

**LIVING ONTOLOGIES:
COLLABORATIVE KNOWLEDGE
STRUCTURING ON THE INTERNET**

by Jenifer Tennison, BSc

Thesis submitted to the University of Nottingham for the degree of
Doctor of Philosophy, May 1999

THESIS CONTAINS

CD

CONTENTS

CONTENTSII

ABSTRACT..... IX

ACKNOWLEDGEMENTSX

CHAPTER 1 SUMMARY 1

1.1 Purpose and Scope 1

1.2 The APECKS System.....2

1.3 Thesis Outline2

1.3.1 Part One: Background.....2

1.3.2 Part Two: APECKS2

1.3.3 Part Three: Future Work.....3

CHAPTER 2 KNOWLEDGE ENGINEERING.....4

2.1 Summary4

2.2 Introduction5

2.3 Ontologies6

2.4 Classical and Constructivist Approaches7

2.5 Knowledge Acquisition9

2.5.1 Techniques.....10

2.5.1.1 Laddered grids.....11

2.5.1.2 Card sorts.....11

2.5.1.3 Repertory grids12

2.5.2 The KA Bottleneck14

2.5.2.1 Multiple Experts.....15

2.5.2.2 Direct KA17

2.5.2.3 Incidental KA18

2.6 Knowledge Sharing.....21

2.6.1 Accessing Ontologies.....22

2.6.1.1 Ontology Servers.....22

2.6.1.2 Corporate Memory Systems24

2.6.2 Collaboration.....25

2.6.2.1 Communication in Ontology Servers.....25

2.6.2.2 Combining Ontologies.....27

2.7 Knowledge Modelling28

2.8 Conclusions31

CHAPTER 3 DISTRIBUTED COMPUTER-MEDIATED COMMUNICATION 34

3.1 Summary 34

3.2 Introduction 35

3.3 Computer-Mediated Communication 35

 3.3.1 Collaborative Virtual Environments 37

 3.3.2 Describing CMC Systems 39

 3.3.2.1 Channels 39

 3.3.2.2 Functions 41

3.4 Multi-User Domains 42

 3.4.1 Classification 44

 3.4.1.1 Channels 44

 3.4.1.2 Functions 44

 3.4.2 Advantages and Disadvantages 46

 3.4.2.1 Accessibility 46

 3.4.2.2 Learnability 46

 3.4.2.3 Archivability 47

 3.4.2.4 Persistency 47

 3.4.2.5 Authorability 47

 3.4.2.6 Programmability 48

 3.4.2.7 Reusability 48

 3.4.3 WWW-enhancement 48

3.5 Design Rationale Systems 50

 3.5.1 Issue-Based Information System (IBIS) 51

 3.5.2 Procedural Hierarchy of Issues (PHI) 54

 3.5.3 Design Representation Language (DRL) 56

 3.5.4 Questions, Options and Criteria (QOC) 58

3.6 Conclusions 60

CHAPTER 4 INFORMATION RETRIEVAL 63

4.1 Summary 63

4.2 Introduction 64

4.3 Hypertext Systems 64

 4.3.1 Problems with Hypertext 66

 4.3.1.1 Cognitive Overhead 66

 4.3.1.2 Disorientation 67

 4.3.2 The World Wide Web 68

 4.3.2.1 Standards 68

 4.3.2.2 Browsers 70

- 4.4 Navigation and Mapping 71
 - 4.4.1 Schematic Representations 71
 - 4.4.2 Spatial Representations 74
- 4.5 Browsing and Searching 74
 - 4.5.1 Information Retrieval on the World Wide Web 75
 - 4.5.1.1 Classified Directories 75
 - 4.5.1.2 Automated Search Engines 76
- 4.6 Collaborative Information Retrieval 77
 - 4.6.1 Single-User Searches 77
 - 4.6.2 Multi-user Searches 78
- 4.7 Conclusions 79
- CHAPTER 5 APECKS SYSTEM REQUIREMENTS AND DESCRIPTION..... 82**
 - 5.1 Summary 82
 - 5.2 Introduction 83
 - 5.3 System Requirements 85
 - 5.3.1 Assumptions 85
 - 5.3.1.1 Roles in APECKS 86
 - 5.3.1.2 Change in APECKS 86
 - 5.3.1.3 APECKS in Use 87
 - 5.3.2 Usage Lifecycle 87
 - 5.3.3 Functional Requirements 88
 - 5.3.3.1 Seeding 88
 - 5.3.3.2 Role Construction 90
 - 5.3.3.3 Role Comparison 90
 - 5.3.3.4 Discussion 91
 - 5.3.3.5 Change Tracking 91
 - 5.3.4 Technical Requirements 92
 - 5.4 System Description 93
 - 5.4.1 Architecture 93
 - 5.4.2 Knowledge Representation 95
 - 5.4.3 User Interface 96
 - 5.4.3.1 Connecting to APECKS 96
 - 5.4.3.2 Objects and Pages 96
 - 5.4.3.3 Interaction Mechanisms 97
 - 5.4.4 Construction Support 101
 - 5.4.4.1 Internal Knowledge Acquisition Support 102
 - 5.4.4.2 External Knowledge Acquisition Support: WebGrid 103

5.4.4.3 Consistency Checking.....	107
5.4.5 Comparison Support	114
5.4.6 Discussion Support	120
5.4.6.1 Evaluative Criteria.....	121
5.4.6.2 Rationale.....	121
5.4.6.3 Free Annotation.....	122
5.5 Conclusions	122
CHAPTER 6 SCENARIOS.....	124
6.1 Summary	124
6.2 Construction: APECKS as an Ontology Server	125
6.2.1 Background	126
6.2.2 Creating a Domain	127
6.2.3 Creating Individuals in a Domain	128
6.2.4 Creating Classes	129
6.2.5 Creating Subclass Partitions	130
6.2.6 Creating Slots with Default Values	130
6.2.7 Creating Slots with Inverses	131
6.2.8 Creating Rationales	132
6.3 Construction: APECKS as a Learning Tool.....	135
6.3.1 Background	136
6.3.2 Creating a Role.....	136
6.3.3 Using Knowledge Elicitation Prompts.....	137
6.3.4 Creating Subclass Partitions	137
6.3.5 Classifying Individuals	138
6.3.6 Providing Explanations for Ignored Prompts	139
6.3.7 Using WebGrid-II for Elicitation	140
6.3.8 Changing the Names of Slots	141
6.3.9 Limiting the Range of a Slot for a Class	142
6.3.10 Reducing Inconsistencies.....	144
6.3.11 Changing Categorical Slots to Subclass Partitions.....	146
6.4 Comparison: APECKS as a Research Tool.....	147
6.4.1 Background	148
6.4.2 Creating Multiple Roles.....	148
6.4.3 Comparing Class Hierarchies	150
6.4.4 Comparing Roles using WebGrid-II.....	150
6.4.5 Searching Annotations.....	152
6.5 Discussion: APECKS as a Decision Support System.....	152

6.5.1 Background	153
6.5.2 Creating Criteria.....	154
6.5.3 Creating 'Consensual' Roles.....	155
6.5.4 Creating Additional Individuals	155
6.5.5 Searching for Changes	155
6.5.6 Multiple Classifications	156
6.5.7 Discussing Differences between Roles	156
6.6 Conclusions	157
CHAPTER 7 APECKS EVALUATION	158
7.1 Summary	158
7.2 Introduction	159
7.3 Method.....	160
7.3.1 Subjects.....	160
7.3.2 Design.....	160
7.3.3 Materials	161
7.3.4 Equipment	162
7.3.5 Procedure.....	162
7.4 Results	163
7.4.1 System Usability	163
7.4.1.1 Time Series Analysis	163
7.4.1.2 Usability Scale.....	165
7.4.1.3 Comments.....	165
7.4.2 Ontology Qualities	169
7.4.2.1 Quantitative Measures	169
7.4.2.2 Qualitative Scale	172
7.4.2.3 Quantitative & Qualitative Correlations	172
7.4.2.4 Seed Ontologies & Subjects' Ontologies	173
7.4.2.5 Task Effects	173
7.4.2.6 Background Effects.....	174
7.4.3 Protocol Analysis	174
7.4.3.1 Time per Page	174
7.4.3.2 Forms submitted.....	175
7.4.3.3 Ownership	175
7.4.3.4 Object Type	176
7.4.3.5 WebGrid-II	177
7.4.3.6 Prompts.....	178
7.4.3.7 Sequence Analysis	179

7.5 Discussion183

 7.5.1 System Usability183

 7.5.2 Ontology Qualities184

 7.5.3 Protocol Analysis185

7.6 Conclusions185

CHAPTER 8 FUTURE WORK.....187

8.1 Summary187

8.2 Introduction188

8.3 Evolution of APECKS.....188

 8.3.1 Knowledge Representation.....189

 8.3.1.1 Frames189

 8.3.1.2 Facets190

 8.3.1.3 Hierarchies.....190

 8.3.1.4 Axioms191

 8.3.1.5 Including Ontologies.....192

 8.3.2 MOO Integration.....192

 8.3.2.1 Virtual Environment192

 8.3.2.2 Synchronous Communication195

 8.3.3 Network Integration195

 8.3.4 Versioning Support196

 8.3.5 Extended Evaluation197

8.4 Theoretical Research.....197

 8.4.1 Ontology Design198

 8.4.2 Comparing and Combining Ontologies199

 8.4.3 Discussion Formalisms.....200

8.5 Recent Developments201

 8.5.1 XML201

 8.5.2 Schemas.....202

 8.5.3 Styling Languages.....205

 8.5.4 Linking Languages.....206

8.6 Conclusions209

8.7 Final Word.....212

REFERENCES216

APPENDIX I APECKS README FILE..... I

APPENDIX II BACKGROUND QUESTIONNAIRE..... VI

APPENDIX III USABILITY QUESTIONNAIRE VIII

APPENDIX IV ONTOLOGY QUESTIONNAIRE XII

APPENDIX V VOCABULARY LIST XIII

APPENDIX VI INSTRUCTIONS – FIRST SESSION XV

APPENDIX VII INSTRUCTIONS – LATER SESSIONS XVI

APPENDIX VIII MAMMALS XVIII

APPENDIX IX LACEPEDE, 1799 XIX

APPENDIX X SIMPSON, 1945 (EXTANT GROUPS ONLY) XX

APPENDIX XI BENTON, 1990 XXII

APPENDIX XII CORBET & HILL, 1991 XXIII

APPENDIX XIII ZOOGEOGRAPHICAL REGIONS XXIV

ABSTRACT

This thesis discusses the issues involving the support of Living Ontologies: collaborating in the construction and maintenance of ontologies using the Internet.

Ontologies define the concepts used in describing a domain: they are used by knowledge engineers as reusable components of knowledge-based systems. Knowledge engineers create ontologies by eliciting information from domain experts. However, experts often have different conceptualisations of a domain and knowledge engineers often have different ways of formalising their conceptualisations.

Taking a constructivist perspective, constructing ontologies from multiple conflicting conceptualisations can be seen as a design activity, in which knowledge engineers make choices according to the context in which the representation will be used. Based on this theory, a methodology for collaboratively constructing ontologies might involve comparing differing conceptualisations and using these comparisons to initiate discussion, changes to the conceptualisations and the development of criteria against which they can be evaluated.

APECKS (Adaptive Presentation Environment for Collaborative Knowledge Structuring) is designed to support this methodology. APECKS aims not only to support the collaborative construction of ontologies but also to use ontologies to present information to its users adaptively within a virtual environment. It demonstrates a number of innovations over conventional ontology servers, such as prompted knowledge elicitation from domain experts, automated comparisons between ontologies, the creation of design rationales and change tracking.

A small evaluation of APECKS has shown that it is usable by domain experts and that automated comparisons between ontologies can be used to initiate alterations, investigations of others' conceptualisations and as a basis for discussion. Possible future development of APECKS includes tighter integration with a virtual environment and with other networked knowledge-based tools. Further research is also needed to develop the methodology on which APECKS is based, by investigating ways of comparing, combining and discussing ontologies.

ACKNOWLEDGEMENTS

I would firstly like to thank my supervisor, Nigel Shadbolt. Nigel has given me the freedom to explore many diverging areas, but drawn back my focus when necessary. He has given me sympathy and encouragement to get me through the lows, but stopped me from wallowing in self-pity. And he has identified the areas on which I needed to work, without making me feel criticised. It is impossible to overstate how much I appreciate the time, consideration and support that he has given me over the past four years.

I would also like to thank the other members of staff in the School of Psychology who have all at one time or another been helpful, encouraging or just plain interested: Claire O'Malley, Peter Cheng, David Clarke, Howard Martin, Andrew Derrington and Robin Stevens.

I would very much like to thank Brian Gaines for allowing me to use WebGrid-II within APECKS while it was still in development; for restarting it promptly every time I managed to crash it; and for showing enthusiasm about my research.

Will Trehwella, Tim Brailsford and others in the School of Life Science were invaluable in their support for the evaluation of APECKS. They took time to discuss the study and helped me find volunteers to take part, and I am very grateful for that help. I am also very grateful to Kieron O'Hara, who provided knowledge engineering experience for the study.

The MOOtiny community and other MOOers opened up the Internet to me and encouraged the development of programming experience without which APECKS could not have been developed. I would particularly like to thank Daniel K. Schneider, Sam Latt Epstein and Kristian Fuglevik. Virtual Environments International, specifically Barry Hardy, Stephen Doughty and Martin Paretti, and CoMentor, specifically Graham Gibbs, Catherine Skinner and Andrew Teal, used APECKS in the real world and gave me invaluable feedback for its development.

I would like to thank those of my friends with whom I have discussed many of the ideas that are in this thesis, namely Dave Snowdon, Elizabeth Churchill, Louise Crow, Jolanda Tromp and Gavin Bell.

I would also like to thank my family, Barry, Mimi and Imogen, for listening, sympathising, encouraging, inspiring and admonishing me at the appropriate times. I particularly appreciate how enthusiastic Barry has been about new ideas and how unjudgemental Mimi has been no matter what.

Finally, Ben and Hedge have been both comforting and distracting. Most importantly, Bill has given me constant unconditional and uncomplaining support, both materially and emotionally. He has always been there for me when I needed him: understanding, affectionate and altruistic.

This research was funded by the School of Psychology, University of Nottingham.

CHAPTER 1 SUMMARY

1.1 PURPOSE AND SCOPE

This thesis addresses the problem of how to support collaborative distributed ontological engineering.

Ontologies are explicit conceptualisations of a domain at the knowledge level. They specify the concepts that are used to reason about a domain. Ontologies are reusable: knowledge engineers can construct different knowledge-based systems for different contexts around the same ontology. Ontologies also facilitate communication and can be used to give inter-operability across different systems. Ontological engineering involves constructing and maintaining ontologies. Ontologies are constructed based on knowledge elicited from experts in a domain through a process called knowledge acquisition.

During knowledge acquisition, ontological engineers elicit knowledge from multiple domain experts to give ontologies greater breadth and depth. The multiple sources of knowledge are never completely consensual: different experts use different terminology or think about concepts in different ways. Where experts disagree, ontological engineers have to make decisions about which term or conceptualisation should be used within the ontology. Ontological engineering is thus a design activity.

Distributed ontological engineers often collaborate in the construction of ontologies. Ontology servers support the construction of ontologies and communication about them by making them available on the Internet. However, most ontology servers do not explicitly support the design process: they do not support the creation of rationales by ontological engineers to record why ontologies were designed in the way they were. Nor do they provide a means to choose between different possible designs.

This thesis addresses the problem of supporting collaborative distributed ontological engineering by focusing on supporting the process of design. By taking a constructivist view of ontological engineering, it theorises that ontology servers will be more useful if they explicitly support the exploration of the design space of ontologies by maintaining diversity and encouraging discussion on their relative merits.

A methodology suggested in this thesis for supporting the collaborative design of ontologies involves comparing different ontologies within the same domain to highlight the different options for design. It also draws on design rationale research to formalise the evaluation of ontologies against criteria arising from the context in which the ontology will be used. This thesis also touches on support for communication and information retrieval from both ontologies themselves and archived communication about them.

1.2 THE APECKS SYSTEM

The focus of this thesis is a system that was constructed to support collaborative distributed ontological engineering. The system is named APECKS for Adaptive Presentation Environment for Collaborative Knowledge Structuring. APECKS aims to not only support the collaborative structuring of knowledge into ontologies but also to use ontologies to present information to its users in an adaptive way within a virtual environment.

APECKS is used over the World Wide Web (WWW), as are most ontology servers. It supports collaborating ontological engineers by facilitating the construction, comparison and discussion of ontologies. It also supports the construction of ontologies directly by domain experts and can be used in this way as a corporate memory system for storing organisational information.

1.3 THESIS OUTLINE

This thesis is divided into three sections.

1.3.1 PART ONE: BACKGROUND

The first section contains the first three chapters, and specifies the problem addressed by this thesis. It also introduces background information from relevant research areas.

Chapter 2 describes the problem that this thesis addresses, namely the support for collaborative ontological engineering. It reviews the current state of the art in support for ontological engineers on the Internet. It also discusses some of the methodologies that are used in knowledge acquisition, particularly focusing on acquiring knowledge from multiple experts and directly from experts.

The central aspect of collaboration is communication. Chapter 3 discusses the issues surrounding support for collaborative work through computer-mediated communication. It focuses on two particular technologies: text-based collaborative virtual environments and design rationale systems.

Collaborative ontological engineering results in two types of hypertextual structures: the ontologies themselves and the archived communication about them. Chapter 4 discusses hypertext systems, including the World Wide Web. It focuses on the problems of navigation and information retrieval from hypertext structures.

1.3.2 PART TWO: APECKS

The second section consists of three chapters and describes the APECKS system and its evaluation.

Chapter 5 firstly discusses the requirements for APECKS arising from the discussion in Part One. It goes on to describe how APECKS fulfils those requirements in its internal knowledge representation and its support for the construction, comparison and discussion of the ontologies it holds.

Chapter 6 walks through four scenarios in which APECKS is used. The first two scenarios focus on the construction of ontologies using APECKS, the first by expert knowledge engineers and the second by a naïve domain expert. The third scenario focuses on the comparison of knowledge structures using APECKS. The fourth focuses on the discussion support within the system.

Chapter 7 presents the results of an evaluation of APECKS in which domain experts used the system to construct, compare and discuss mammalian taxonomies. The evaluation demonstrates that domain experts can construct ontologies directly and that APECKS encourages the sharing of points of view.

1.3.3 PART THREE: FUTURE WORK

The final chapter, Chapter 8, discusses some of the areas that future work might address. It firstly outlines the improvements that could be made to APECKS and its further evaluation. It then discusses some of the methodological issues that need further research, such as the identification of evaluative criteria for ontologies, how ontologies can be compared and combined and what formalisms would best support discussion that involves ontologies. Finally, it discusses the impact of recent developments on the design of systems like APECKS in the future.

CHAPTER 2 KNOWLEDGE ENGINEERING

2.1 SUMMARY

This chapter has three purposes. Firstly, it describes the problem that this thesis addresses, namely the requirement for Living Ontologies. Secondly, it discusses some of the solutions or partial solutions that have been proposed to aid ontology construction and maintenance. Finally, it gives some background information about knowledge engineering methodologies that are referred to later in the thesis.

One of the major tasks facing knowledge engineers currently is the creation of reusable knowledge, or ontologies. In ontological engineering, knowledge engineers create explicit representations of (usually consensual) knowledge about a domain. In this chapter I argue that consensus should not be forced on domains. Knowledge engineers should take a constructivist approach to the creation of ontologies, recognising that ontology construction is a design process. This theory implies that ontological engineering involves mapping a design space of alternative ontologies. It also implies that knowledge engineering, the construction of knowledge based applications, involves the selection of ontologies from such a design space.

I therefore introduce the concept of a Living Ontology, one which, like a Living Document, grows and evolves over time as a diverse set of people contribute to it. This chapter discusses three important areas for ontological engineering:

- **Knowledge Acquisition:** considering how expertise from multiple sources can be combined, and how knowledge can be elicited directly from experts or the artefacts they produce.
- **Knowledge Modelling:** the types of representations that can be made of knowledge.
- **Knowledge Sharing:** the support currently available for sharing and communicating about ontologies.

From this discussion, four requirements are generated that need to be fulfilled within applications designed to support Living Ontologies:

1. Support for the construction and development of evolving ontologies
2. An internal representation that can be translated for use by other applications
3. The comparison of several coexistent ontologies within the same domain
4. Support for communication between those constructing the ontology

These requirements are addressed in the design of APECKS, a system supporting Living Ontologies that is described in Chapter 5. The fourth requirement, support for communication, is discussed in detail in the next chapter.

2.2 INTRODUCTION

The process of designing, constructing and maintaining Knowledge Based Systems (KBSs) is known as Knowledge Engineering. Early knowledge engineering focussed on the production of relatively small expert or decision support systems. These systems could be used instead of or in conjunction with an expert in a specific domain to solve particular tasks. Problems arose on three fronts, however: these first generation systems were hard to maintain and their code was not reusable across systems; the methods used to construct them did not scale to larger systems; and they could not supply the levels of explanation demanded by those using them (Shadbolt, 1989).

In 1982, Newell identified the problem with first generation systems: they were built with no distinction between the *symbol level* (the code) and the *knowledge level* (information about the domain itself). The consequent hotchpotch was the cause of the difficulties in maintenance and reuse of the KBSs. By explicitly separating the two levels, Newell theorised, both the code itself and the knowledge on which it was based could be reused in later systems, cutting the work required by the knowledge engineer dramatically.

Newell's ideas were taken on for the so-called second generation expert systems, which sought to model experts rather than transfer their knowledge into code (David, Krivine & Simmons, 1993). Increasingly, knowledge engineering methodologies have been based around the creation of *ontologies*, which give representations of knowledge in a particular domain, and *problem-solving methods* (PSMs), which give representations of how to solve particular tasks, regardless of domain. The intention is that knowledge engineering can simply entail fitting the correct problem-solving method onto the domain ontology. This shifts the emphasis from the creation of KBSs themselves onto the creation of ontologies and PSMs, the former being known as the process of *Ontological Engineering*.

In this chapter, Section 2.3 outlines what ontologies are. Section 2.4 then describes the approaches that can be taken to knowledge and ontological engineering and argues for a constructivist approach to be taken. I then go on to describe three important aspects of ontological engineering and the problems associated with them. These areas are:

- **Knowledge Acquisition:** eliciting knowledge from experts (Section 2.5)
- **Knowledge Sharing:** exchanging knowledge between knowledge engineers (Section 2.6)
- **Knowledge Modelling:** creating representations of knowledge (Section 2.7)

The conclusion (Section 2.8) indicates the issues concerning collaborative distributed ontological engineering that I believe are currently under-supported in ontology servers and outlines the approach of 'Living Ontologies'.

2.3 ONTOLOGIES

Uschold (1998) describes the current usage of the term 'ontology' as meaning:

Short Definition An *explicit* account or representation of some part of a CONCEPTUALISATION (adapted from Guarino & Garetta (1995)).

An ONTOLOGY may take a variety of forms, but necessarily it will include a vocabulary of terms, and some specification of their meaning. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the DOMAIN and constrain the possible interpretations of terms.

An ONTOLOGY is virtually always the manifestation of a shared understanding of a DOMAIN that is agreed between a number of agents. Such agreement facilitates accurate and effective communication of meaning, which in turn leads to other benefits such as inter-operability, reuse and sharing.

In the third paragraph quoted above, Uschold states that ontologies are usually explicit representations of a consensual conceptualisation. This is certainly true in definitions of ontologies constructed using the classical approach outlined in Section 2.4. Under the constructivist approach, it is recognised that the consensual nature of an ontology is limited by the experts who have contributed to it. While part of that ontology may be consensual with a different set of experts, part of it may not be.

According to van Heijst, Schreiber & Wielinga (1997), ontologies define the structure and vocabulary of domain knowledge. Thus, having received an ontology, a knowledge engineer can adapt it for a particular context by inserting facts relevant to that context. However, ontologies can also contain domain knowledge to a greater or lesser extent. An ontology meant purely for KBS development may still include domain knowledge in the way of examples of concepts, for example.

Uschold goes on to identify three dimensions along which ontologies may vary:

- **Formality:** the representation method used to describe the ontology. These are discussed in Section 2.7.

- **Purpose:** the reason for the existence of the ontology. Uschold identifies three reasons:
 - **Communication** between people.
 - **Inter-operability** among systems.
 - **Systems engineering benefits** for reusability, reliability, specification and KA.
- **Subject Matter:** the area the ontology covers. Uschold characterises four types:
 - **Domain ontologies** (e.g. medicine, geology, finance)
 - **Upper models** (general world knowledge)
 - **Task, method or problem solving ontologies**
 - **Representation ontologies or meta-ontologies** (e.g. the Frame Ontology)

Ontologies used for different purposes stress different aspects in their design. Ontologies built in order to facilitate communication between people need to be less formal than those used for inter-operability or systems engineering (see Section 2.7). They may also be concerned with the representation of concepts in different languages, as in the Generalized Upper Model (Bateman, Magnine & Rinaldi, 1994).

The subject matter of an ontology also affects how it is designed and used. Domain ontologies are increasingly being generated and used by people without a knowledge engineering background (this is discussed later in relation to the generation of XML schemas in Section 8.5.2). Upper models are designed to be included within, or to form the basis of, other ontologies while meta-ontologies are used to define the concepts used in knowledge representation (see Section 2.7).

2.4 CLASSICAL AND CONSTRUCTIVIST APPROACHES

The task of Ontological Engineers is to construct ontologies. To support this task, they construct systems, several examples of which are given in this chapter, and develop methodologies (see Jones, Bench-Capon & Visser (1998) for a good overview). The systems and methodologies used can be classed in terms of two extremes, distinguished by the underlying philosophy or theory that dictates their design. At the Classical extreme, ontological engineers take a Platonic view and presuppose that there is a single, true ontology. At the Constructivist extreme, ontological engineers assume that there are multiple, equally valid, ontologies, with varying advantages and disadvantages depending on the situation in which they are to be used. For example, Fridman Noy & Hafner (1997) give a good constructivist analysis of various prominent ontologies (mostly upper models).

As we shall see in Section 2.5.2.1 below, ontologies are based on the expertise of several experts, including those contributing indirectly through books. Multiple experts are required to gain the

coverage necessary for an ontology, but, more often than not, experts do not agree with each other. For classicists, the ontological engineering process involves discovering the underlying consensual ontology from these diverse viewpoints. For constructivists, the assorted accounts of the domain provide a rich *design space* of possible ontologies. To them, the ontological engineering process involves exploring the design space itself and the impact of design choices on the utility of the ontologies produced.

The distinction between classical and constructivist approaches is perhaps clearest when considering the question of whether ontologies are task dependent or not. As Chandrasekaran, Josephson & Benjamins (1998) point out, the aspects of reality that are made explicit within an ontology depend on the task conceived of during its construction, even if the reality itself is not task-dependent. From the classical approach, the ultimate goal is the production of an ontology in which all aspects of reality are explicit. Since these ontologies would be unmanageably large if used as a whole, methods of identifying those concepts that are of importance to a particular task are used to generate subsets that can be used in KBSs. For example, in Swartout et al. (1996), seed concepts in the domain of military air campaign planning were used to identify subtrees of the large-scale SENSUS ontology, which were then combined into a domain-specific ontology. Similar techniques could be used to generate *task-specific* ontologies from larger *domain-specific* ontologies.

The constructivist approach to the same problem is to identify and make explicit the task-dependent assumptions of a particular ontology. These assumptions can then be matched to an appropriate problem-solving method (Fensel & Benjamins, 1996). However, task-related assumptions are not the only type implicit in ontologies. *Ontological commitments* are assumptions made by an ontological engineer about both how knowledge should be modelled (see Section 2.7) and how the domain itself should be structured. Skuce (1995) recognises the need for specifying the assumptions used in constructing an ontology. He recommends that, for each category, the following types of assumption are made explicit:

- **Conceptual:** the rationale underlying the category and why it should be included in the ontology
- **Terminological:** the terms that can be used for the category, a single principal term and the reasons behind choosing that term
- **Definitional:** a dictionary-like definition of the concept

Taking the constructivist viewpoint, Ontological Engineering can be seen as a process of *collaborative design*. Ontologies are conceived of as resources that are constructed by groups of knowledge engineers. Once produced, they should be made generally available to knowledge engineers in libraries or repositories in order to fulfil one of the purposes of ontologies: to provide

reusability of domain knowledge across systems. Ontologies are also seen as changing resources that are expanded over time to reflect developments in their domains. In each of these situations, there is a need for collaboration between those constructing, using and maintaining the ontology.

A mixture of the two approaches is evident in Cyc®¹, an upper model seeking to represent common-sense knowledge. Within Cyc, *contexts* or *micro-theories* are used to both partition the knowledge base and simplify assertions (Lenat, 1998). Contexts make explicit the assumptions that lie between a set of assertions, which allows opposing assertions to be made within different contexts. The types of assumptions made within contexts can be classified in terms of 12 dimensions: Absolute Time, Type of Time, Absolute Place, Type of Place, Culture, Sophistication/Security, Topic/Usage, Granularity, Modality/Disposition/Epistemology, Argument-Preference, Justification and Let's. This shows a constructivist recognition of the importance of making explicit the assumptions made within an ontology. However, differences between ontologies do not simply reside in the assertions that are made within them, but also in the way in which they are structured and the terms that are used. Such differences are not captured within Cyc.

From this argument, systems and methodologies to support ontological engineering can and should borrow from the tools and techniques used in collaborative design in other areas. Design rationale systems (see Section 3.5), for example, support the exploration of a design space and the documentation of design decisions by formalising the process of design. They also support collaborative work both by directing communication between collaborators and by recording this communication for future designers.

This thesis takes a constructivist approach to ontological engineering. It adopts collaborative techniques in knowledge engineering (this chapter), computer-mediated communication (Chapter 3) and information retrieval (Chapter 4) to develop a methodology to support collaborative ontological engineering, which is then instantiated in a prototype system (Chapter 5).

2.5 KNOWLEDGE ACQUISITION

The first problem that Knowledge Engineers face when attempting to construct a Knowledge-Based System (KBS) is Knowledge Acquisition (KA). KA involves eliciting knowledge from experts and from other resources. Knowledge acquired from experts is then formalised using the knowledge modelling techniques discussed in Section 2.7, and coded into a KBS. Section 2.5.1 gives some background about Knowledge Acquisition techniques that are available to knowledge engineers. Section 2.5.2 discusses the practical problems involved in carrying out KA and some of the solutions that are used to combat them.

¹ More information about Cyc is available at <http://www.cyc.com>

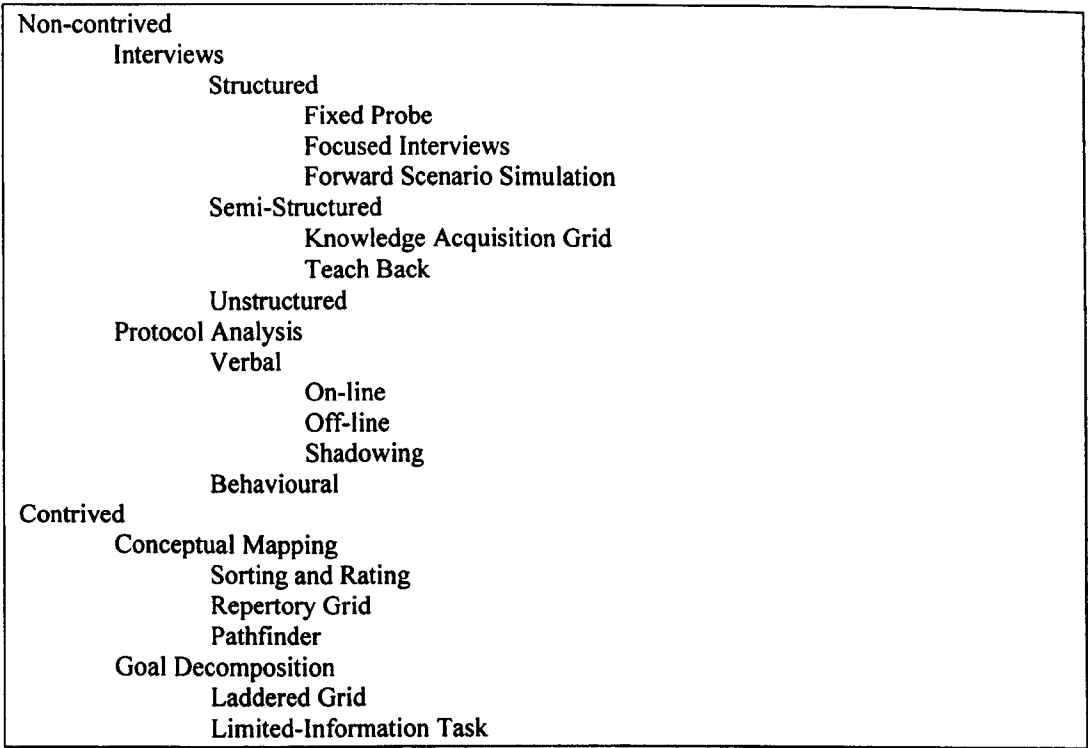


Figure 2.1: A taxonomy of elicitation methods. From Shadbolt & Burton (1995)

2.5.1 TECHNIQUES

Shadbolt & Burton (1995) give a taxonomy of knowledge acquisition techniques, which is shown in Figure 2.1. The highest level division is one between *non-contrived* or *natural methods* and *contrived methods*.

Non-contrived methods involve the expert expressing knowledge in a natural way, such as talking about it in an interview or carrying out a knowledge-intensive task (protocol analysis). The major disadvantage of these methods is that they result in informal and usually unstructured records (depending on the degree of structure used in the interview or protocol analysis). These have to be firstly transcribed and then analysed in order to generate more formal representations of the knowledge expressed by the expert. Each of these activities, conducting the interview, transcribing it and analysing it, requires human involvement, usually a trained knowledge engineer. The feasibility of using non-contrived methods *without* involving a knowledge engineer is discussed in Section 2.5.2.3.

Contrived methods involve experts expressing knowledge in a manner that is usually unfamiliar to them. While experts may feel uncomfortable expressing knowledge in an unfamiliar way, the knowledge yield is high, and sometimes even better than natural methods (Shadbolt & Burton, 1989). Knowledge expressed using these methods is much more formal than that gathered using non-contrived methods, and often result in semi-formal knowledge representations that can be

KE: Is there a key difference that you would allow you to tell the difference between larvikite found in South Africa and the larvikite found in Norway?

E: Not much except the South African tends to show some, what appears to be flow banding.

KE: Is there a more general category that larvikite is an example of?

E: Yes it is a variety of syenite.

KE: What alternative examples of syenite are there apart from larvikite?

E: Some of the rocks in Bradgate Park, near Charnwood Forest, Mountsorrel, are syenites.

KE: And how can you tell that these rocks are syenites?

E: It has this very, very clear, we are going back to the presence of the braderite schillerisation which is the main feature, and very coarse grained.

KE: Is syenite in turn an example of a more general type of rock?

E: It is an intrusive igneous rock.

Figure 2.2: An extract from a laddered grid. From the second release of the Sisyphus III material

used without further analysis. This means that many of these techniques can be readily used *without* the involvement of a knowledge engineer, thus enabling direct knowledge acquisition (see Section 2.5.2.2). Three techniques in particular have this advantage, and are discussed below. These are laddered grids, concept or card sorts and repertory grids.

2.5.1.1 Laddered grids

Laddered grids are used to expand and structure a domain through the interactive construction of a classification hierarchy. The process involves firstly identifying a concept within the domain and then moving around the classification hierarchy using one of the following prompts:

- Can you give examples of *concept*? (moves down the hierarchy)
- What alternative examples of *class* are there to *concept*? (moves across the hierarchy)
- What have *concepts* got in common? (moves up the hierarchy)
- What are *concepts* examples of? (moves up the hierarchy)
- How can you tell it is *concept*? (elicits essential properties)
- What is the key difference between *concept1* and *concept2*? (discriminates concepts)

Figure 2.2 gives an extract from a laddered grid interview from the domain 'igneous rocks' showing examples of some of these prompts.

2.5.1.2 Card sorts

Card sorts are used to generate categories or classes under which concepts within a domain can be classified. The technique involves having representations of the concepts, which may be objects,

pictures or simply cards with the name of the concept written on them (hence the term card sort), and having the expert sort these representations into groups. The expert then names these groups. Sorting is done several times, although each time the expert must decide on a different set of categories: one way to enforce this is to ask for a different number of groups to be made during each sort.

Figure 2.3 shows an example card sort from the domain 'igneous rocks'. The rocks that are being sorted are given in the left-most column, while the groups in which they were placed during the five sorts are indicated by numbers in the other columns. The coding of the numbers is given below the table.

2.5.1.3 Repertory grids

Repertory grids arose from Kelly's (1955) Personal Construct Theory. Repertory grids were developed as tools for knowledge acquisition to aid the elicitation of *entities*, *constructs* and *ratings* within a domain. Entities are concepts within the domain, which can be distinguished from each other by the ratings they are given for the constructs.

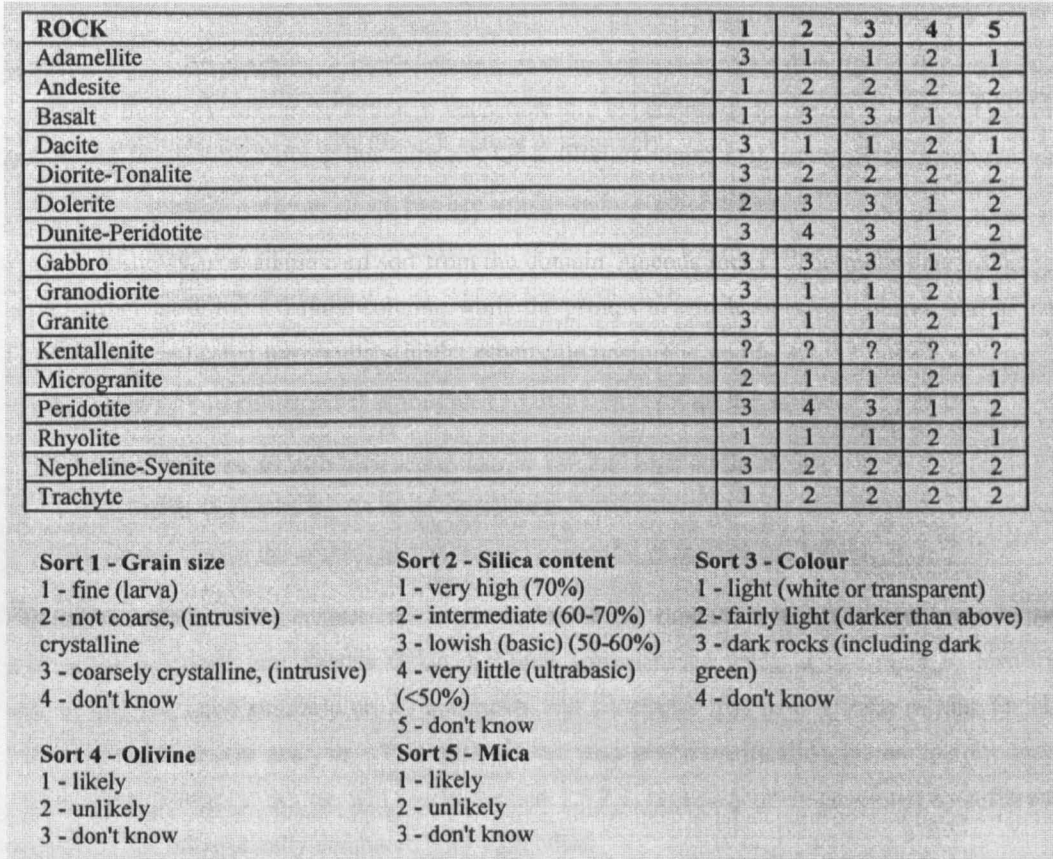


Figure 2.3: A card sort. From the first release of the Sisyphus III material

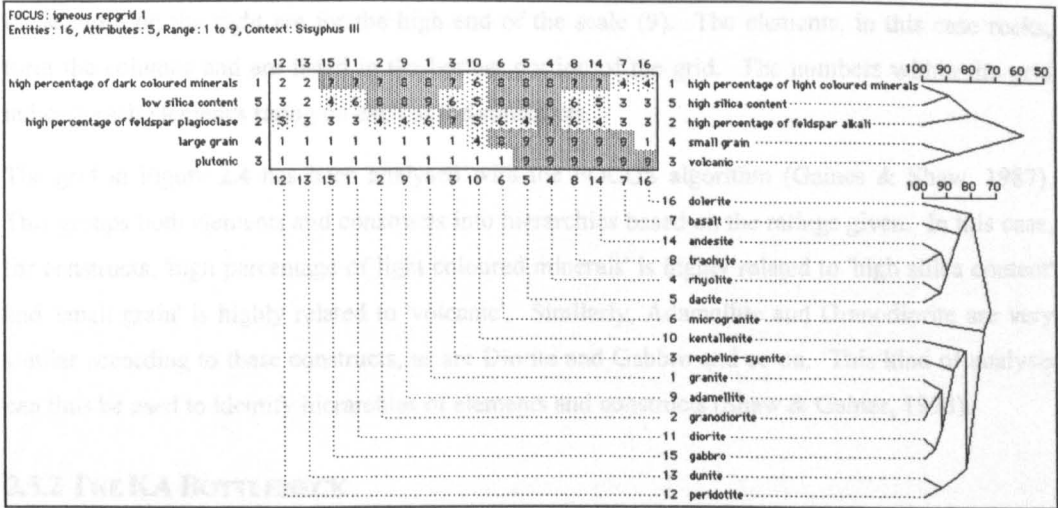


Figure 2.4: A repertory grid. From the first release of Sisyphus III materials.

The process of building a repertory grid is as follows:

1. Identify a starting set of entities, which are usually concrete items within a domain.
2. *Either:*
 - a. Identify three entities from the set, usually those which are currently rated similarly on all constructs, but otherwise through choice or randomly.
 - i) Identify a way in which two are similar and the other different.
 - ii) Name the similarity and difference as two ends of a scale and rate each of the other entities on the scale.
 - or
 - b. Identify two constructs that are given similar values for all the entities.
 - i) Create an entity which scores low on one and high on the other.
 - ii) Rate the entity on the other constructs.
3. Repeat step 2 until the entities and constructs cannot be expanded any longer.

The process of identifying entities and constructs in order to expand the grid is enhanced when the grid is computerised (see Section 2.5.2.2). With computerised grids, it is simple to identify entities that are rated similarly on all constructs and constructs that have similar ratings for all entities through cluster analysis. This analysis can also give classification hierarchies for both entities and constructs. As we will see in Section 2.5.2.1, repertory grids generated by different experts can be automatically compared with each other.

Figure 2.4 shows an example of a repertory grid in the domain 'igneous rocks'. The constructs are listed on the left and right of the grid: the labels on the left are for the low-end of the scale (1)

while those on the right are for the high end of the scale (9). The elements, in this case rocks, form the columns and are listed in the bottom portion of the grid. The numbers within the grid indicate each element's rating on each of the constructs.

The grid in Figure 2.4 has been analysed with the FOCUS algorithm (Gaines & Shaw, 1987). This groups both elements and constructs into hierarchies based on the ratings given. In this case, for constructs, 'high percentage of light coloured minerals' is highly related to 'high silica content' and 'small grain' is highly related to 'volcanic'. Similarly, Adamellite and Granodiorite are very similar according to these constructs, as are Diorite and Gabbro and so on. This kind of analysis can thus be used to identify hierarchies of elements and constructs (Shaw & Gaines, 1998).

2.5.2 THE KA BOTTLENECK

Knowledge Acquisition has been called the 'bottleneck of KBS construction' (Hayes-Roth, Waterman & Lenat, 1983). KA is a time-consuming process and requires a lot of contact with the experts, but, due to their expertise, they have a great deal of other work to do. Not only is it hard to get time with experts, but they are often reticent to take part in the construction of systems that they may fear will replace them and to get involved in a process which has no perceived benefit for them.

There are several partial solutions to the problem of getting hold of experts for KA:

- **Ontologies:** if domain knowledge can be shared, there is no need to carry out KA for each KBS constructed.
- **Multiple Experts:** with multiple experts contributing to a knowledge base, the demand on a single expert is not as great.
- **Direct KA:** if knowledge engineers need not be involved in KA, experts are free to do it whenever they have the time.
- **Incidental KA:** experts produce knowledge artefacts constantly in the reports they write and the forms they fill in. If this knowledge can be mined automatically, they need not even know it happening.

The distinction I make here between *direct* and *incidental* knowledge acquisition is that direct KA involves experts intentionally carrying out knowledge acquisition tasks while incidental KA arises from their normal activity. This distinction is reflected in the distinction between contrived and non-contrived KA techniques discussed in Section 2.5.1. Non-contrived techniques involve the analysis of transcripts from expert interviews or activity, which can be automatically captured (see Section 2.5.2.3), but require additional interpretation. Contrived techniques, on the other hand, are generally well suited to direct use by experts, as we shall see in Section 2.5.2.2.

None of the solutions described here can address all the problems of KA, and even when used in combination, KA will still be a time-consuming process. Section 2.6 below deals with the use of ontologies for KA as well as knowledge management. The other solutions are discussed here.

2.5.2.1 Multiple Experts

The benefits of using multiple experts for KA extend beyond merely being able to distribute the time demands between them. Experts bring their own backgrounds, priorities, assumptions and specialities, giving a KBS the breadth and depth that it needs. Indeed, from the constructivist perspective outlined above, it is essential to have these divergent opinions in order to discover the best ontology for a particular context.

However, when considered from a classical perspective, using multiple experts leads to the problem of how to create a consensual ontology when combining the knowledge from a variety of sources. This problem also occurs when additional perspectives are added later in the development of an ontology, for example adding chemotherapists' expertise to that of surgeons (O'Leary, 1997).

Often, when conflicts arise between conceptualisations, knowledge engineers simply choose which source to use. Obviously, this is a subjective choice, based upon the perceived reliability of the source, and may be susceptible to extraneous political influences rather than the quality of the knowledge itself. The methodologies outlined here attempt to combine knowledge from multiple experts with the maximum of objectivity on the part of the knowledge engineer. Other methods can be found in McGraw & Harbison-Briggs (1989).

Group Elicitation Method

The Group Elicitation Method (GEM) was developed by Guy Boy (1996) to enhance KA carried out in group meetings. GEM is based on the *brainwriting* technique (Warfield, 1971). It seeks to reduce the negative aspects of group elicitation found by Neal & Mantei (1993), namely incomplete consideration of alternatives; unequal participation; lack of a record of the meeting; and dissatisfaction with the process as a whole.

GEM comprises of six stages:

1. The formulation of a statement of the issue that will be discussed at the meeting and the selection of around seven participants for the meeting.
2. The generation (in the meeting) of viewpoints on the issue that was decided on in step 1. Each participant writes a number of viewpoints on a sheet, which is then passed to the person on their right. On receiving another's sheet, a participant can register agreement or disagreement with any of the viewpoints on the sheet or add a new viewpoint of their own.

3. Once the viewpoints are generated, the meeting facilitator (a knowledge engineer) reformulates them into a number of concepts, with some input from the participants.
4. The participants then compare each concept with every other concept and complete a grid on which is marked whether a concept is more, less or equally important than another.
5. Various measurements are then computed for each concept based on the scores that they were given in step 4. These include its mean priority, inter-participant consistency and stability.
6. The results of the analysis are presented to the group and they discuss the consensus.

While GEM is explicitly designed to bring about consensus between the participants, it can also identify areas where there is little consensus. This can be used to direct discussion towards contentious areas.

The issue generation method in GEM is similar, though not as structured, as that seen in Design Rationale systems (see Section 3.5). From the example presented within Boy's (1996) paper, it appears that in design rationale terms, the viewpoints generation step gives a mixture of sub-Issues/Questions and Positions/Options. The similarities may lead to problems in both the identification of the main issue to be discussed (step 1) and the formulation of concepts (step 3). In Design Rationale research, it has been found that the phrasing of Questions has a large impact on the rationale generated (Bellotti, MacLean & Moran, 1991), potentially biasing the knowledge acquisition.

As Boy points out, GEM is similar to the repertory grid technique described in Section 2.5.1.3, in that participants judge concepts on a rating scale. However, unlike repertory grids, participants are assigned the construct (importance) and, instead of rating each concept independently, they simply record the direction of the comparison between two concepts. This means that only general distinctions can be made between concepts.

Repertory Grids and Sociogrids

In 1989, Shaw & Gaines demonstrated how the repertory grid technique (see Section 2.5.1.3) could be extended in order to compare the conceptual structures of different experts. They stated that experts could differ both in the concepts they use and the terminology they use for the concepts. Based on this, there are four relations between (parts of) experts' conceptual systems, which are summarised in Table 2.1. Sociogrids identify how experts' conceptualisations differ according to this classification.

To create sociogrids, experts first agree on a set of entities that are relevant to the domain. This can be done through the separate generation of repertory grids from which a consensual set of entities are identified, or through techniques similar to those used in issue generation in GEM, as explained above. The experts then each complete a repertory grid for the agreed on set of entities.

		Terminology	
		Same	Different
Attributes	Same	Consensus Experts use terminology and concepts in the same way	Correspondence Experts use different terminology for the same concepts
	Different	Conflict Experts use same terminology for different concepts	Contrast Experts differ in terminology and concepts

Table 2.1: Consensus, conflict, correspondence and contrast among experts. From Shaw & Gaines (1989).

The construction of this repertory grid involves only the addition of *constructs*, not of entities, as the entities are already themselves consensual. The repertory grids generated in this way thus represent each expert's particular perspective on the salient features of the entities. Two types of comparisons are then made:

- **Attribute Compare**

Where there are similarities in ratings for different constructs for the same entities across different grids, this indicates *correspondence*: different terminology being used for the same concept. Constructs that cannot be matched from their ratings with those in other grids indicate *contrast*. (This assumes that the experts do not use the same terms for their attributes, in which case consensus or conflict would be indicated.)

- **Entity-Attribute Compare**

The second comparison first involves the experts swapping empty grids. Each of the grids is emptied of ratings and given to every other expert. The experts then rate the entities on another's constructs. Where the ratings given are the same (or similar), this indicates *consensus* between the experts; where they are different, it indicates *conflict*.

The Sociogrid methodology, particularly Attribute Compare, is used to compare ontologies within APECKS, as we shall see in Section 5.4.5 and Section 6.4.4.

2.5.2.2 Direct KA

One of the ways in which the Knowledge Acquisition bottleneck can be ameliorated is through KA directly from the expert, without the intervention of a knowledge engineer. Direct KA has several advantages:

- It allows experts to carry out KA when they have the time rather than in pre-arranged meetings.

- It allows KA to occur without knowledge engineers physically present, which means that remote experts can be used without the extra cost and time of travel.
- It frees knowledge engineers, allowing them to devote more time to the KA methods that cannot be carried out directly by experts.
- It puts less social pressure on the experts, who often feel suspicious of contrived techniques.

However, there are a number of disadvantages:

- Experts are not familiar with the constraints or requirements of knowledge modelling, and without guidance may not produce knowledge in a useful form.
- Without external pressure, experts may not carry out the KA.

The question of whether direct KA can produce useful results is discussed in Chapter 7, on the evaluation of APECKS.

There are an ever-increasing number of computerised KA tools that could be used, in theory, directly by experts (O'Hara, Shadbolt & van Heijst, in press). However, most are designed as aids for knowledge engineers and lack the guidance that domain experts would need should they use them independently. Knowledge Support System Zero (KSS0: Gaines, 1987; Gaines & Shaw, 1987), a repertory grid elicitation and analysis tool, however, has been made available over the World Wide Web as WebGrid (Gaines & Shaw, 1996a; Gaines & Shaw, 1996b) and WebGrid-II (Gaines & Shaw, 1998; Shaw & Gaines, 1998), and has made some concessions to the assistance that naïve users require, by supplying help at each stage (see Figure 2.5).

2.5.2.3 Incidental KA

Knowledge is expressed naturally by people as they work, particularly when working collaboratively in teams. Capturing this knowledge does not require any effort on the part of the experts: they simply continue to carry out their work and communicate with their colleagues. This has particular benefits for the maintenance of knowledge, especially in fast-changing domains, since KA can become an ongoing activity, with knowledge bases adapting to changing circumstances. An example of incidental KA in action is PIERS (Pathology Expert Interpret Report System), which, using Ripple Down Rules, has expanded from 200 original rules up to 1800 simply through everyday use by pathologists (Kang, Compton & Preston, 1998). Incidental KA is of particular use in Corporate Memory Systems, which are discussed in Section 2.6.1.2.

Extracting knowledge from normal activity can be seen as a non-contrived KA method (see Section 2.5.1). It requires that activity be recorded in such a way that it can be analysed and formalised. The issue of capturing activity and communication for later use by others is arguably the central topic of Computer Supported Co-operative Work and is discussed in detail in both Chapter 3 and Chapter 4. In incidental KA, however, the captured activity may not necessarily be

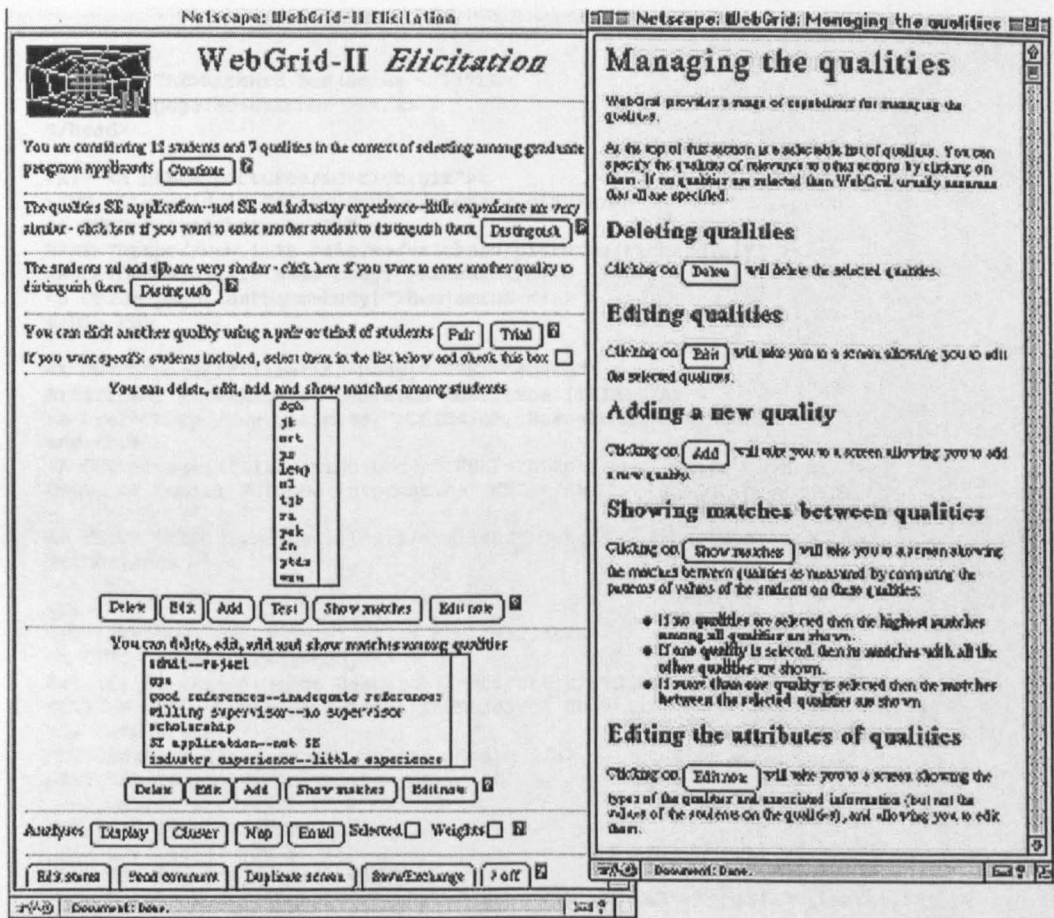


Figure 2.5: Context-sensitive help in WebGrid-II. From Gaines & Shaw (1998).

human-readable, but must be formalised or formalisable. Structured communication methodologies, such as design rationale (see Section 3.5) are therefore of particular relevance.

The growth of the World Wide Web (WWW) has been a revelation to knowledge engineers. This is not simply because of the increased facility for collaboration and knowledge sharing that it brings (discussed in Section 2.6), but because of the vast amount of information that is now accessible in an electronic format. As well as online databases, experts are supplying knowledge in the form of papers, guides, teaching materials and other resources on almost every topic. The ability to retrieve and formalise this knowledge is one of the primary thrusts in knowledge acquisition today.

Two related efforts are being made in this field, both of which are discussed in more depth in Section 8.5.

- **Intelligent Agents**

Intelligent agents, or 'softbots', can be used to gather knowledge from Internet resources. Their primary purpose is to identify and retrieve knowledge for a user, but they can also be

```
<html>
<head><TITLE>Richard Benjamins </TITLE>
<a ONTO="page:Researcher"> </a>
</head>

<H1> <A HREF="pictures/id-rich.gif">
<IMG align=middle SRC="pictures/richard.gif"></A>
<a ONTO="page[photo=href]"
HREF="http://www.iiia.csic.es/~richard/pictures/richard.gif" ></a>
<a ONTO="page[firstName=body]">Richard</a>
<a ONTO="page[lastName=body]">Benjamins </a>
</h1> <p>

<A ONTO="page[affiliation=body]" HREF="#card">
Artificial Intelligence Research Institute (IIIA) </A> -
<a href="http://www.csic.es/">CSIC</a>, Barcelona, Spain <br>
and <br>
<A ONTO="page[affiliation=body]" HREF="http://www.swi.psy.uva.nl/">
Dept. of Social Science Informatics (SWI) </A>
-
<A HREF="http://www.uva.nl/uva/english/">UvA</A>, Amsterdam, the
Netherlands

<DL>
<DT><STRONG><A HREF=".../IIIA.html">IIIA</A> -
<a ONTO="page[address=body]">
Artificial Intelligence Research Institute </STRONG>
<DT><EM>CSIC - Spanish Scientific Research Council</EM>
<DT>Campus UAB
<DT>088193 Bellaterra, Barcelona, Spain </a>
<DT><IMG SRC="gifs/tel.gif">
voice: +34-3-580 95 70
<DT><IMG SRC="gifs/fax.gif">
fax: +34-3-580 96 61
<DT><IMG SRC="gifs/email.gif">
Email:<A HREF="mailto:richard@iiia.csic.es" ONTO="page[email=href]">
richard@iiia.csic.es</A>
<DT>URL: <A HREF="http://www.iiia.csic.es/~richard/">
http://www.iiia.csic.es/~richard</A>
</DL></font>

</body>
</html>
```

Figure 2.6: Example web page annotated with the ONTO attribute. From Benjamins & Fensel (1998).

used to construct ontologies based on the information they find, as in IMPS (Crow & Shadbolt, 1998). Intelligent agents can be domain-specific, with a domain ontology hard-wired into the system, as in Ontobroker for the domain of knowledge acquisition (Fensel et al., 1998). Alternatively, they can be domain independent, constructing the ontology they need on-the-fly, as in IMPS (Crow & Shadbolt, 1998).

- **Formalisation of Information**

While natural language analysis can construct knowledge representations from plain text, it is a lot easier to identify the significance of a resource or section of a resource if it is made explicit within the resource itself. The KA² initiative (Benjamins & Fensel, 1998) uses tags within WWW pages to give meta-information about the content. For example, the page shown in Figure 2.6 specifies that the person represented by the page is a Researcher, with a

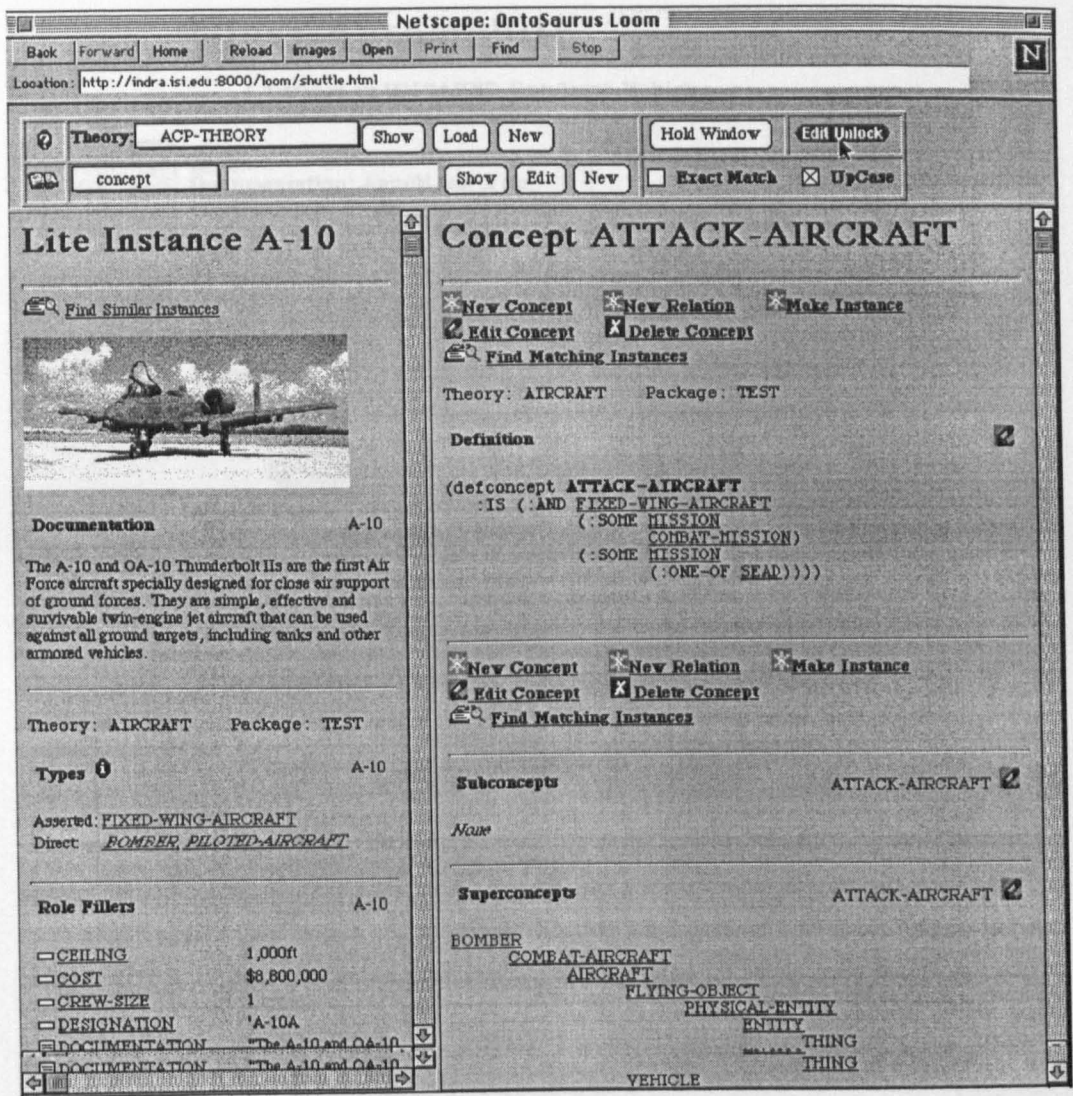


Figure 2.7: The Ontosaurus Browser. From Swartout et al. (1996).

photo at <http://www.iiia.csic.es/~richard/pictures/richard.gif>, the first name 'Richard' and last name 'Benjamins', affiliated with 'Artificial Intelligence Research Institute (IIIA)' and so on. The development of XML (eXtensible Markup Language) and RDF (Resource Description Framework) give standard way in which to add this type of information to WWW pages. This recent development and its implications are discussed in detail in Section 8.5.

2.6 KNOWLEDGE SHARING

Probably the area that is seeing most activity in the effort to reduce the need for knowledge acquisition is that of knowledge sharing and reusability of knowledge. In this section, I discuss several aspects of these activities, focusing on the use of ontologies for sharing domain knowledge. Section 2.6.1 discusses systems that have been used to share ontologies, both for

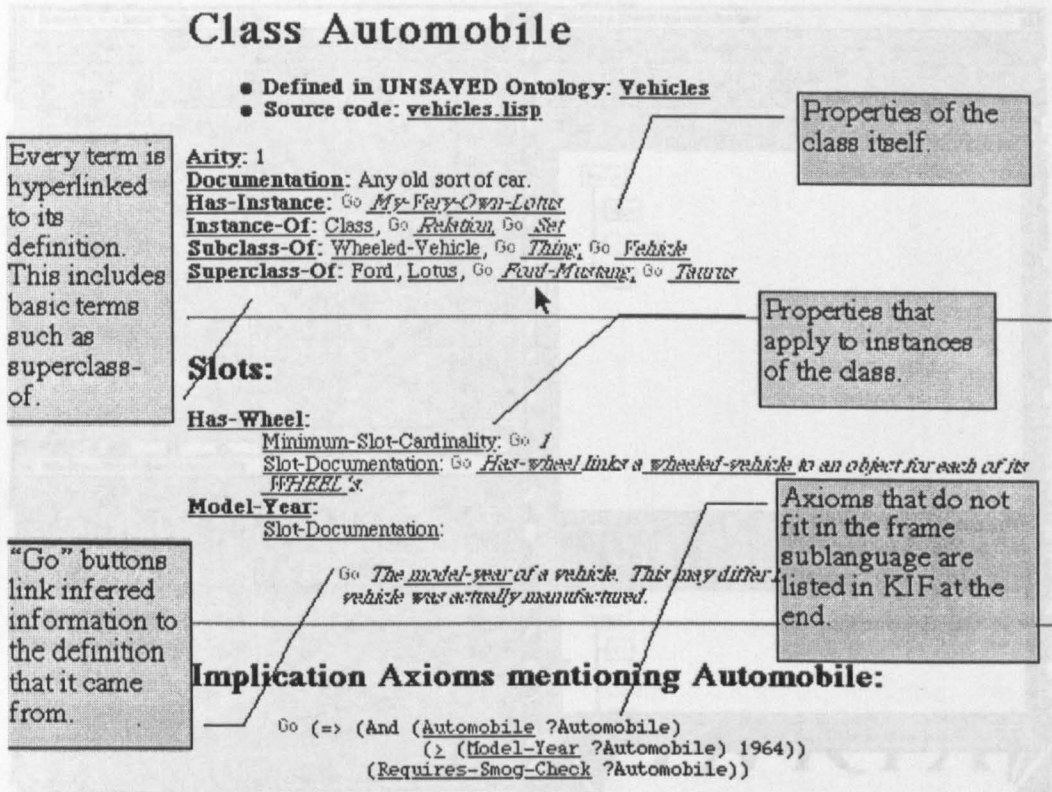


Figure 2.8: A frame-based view of the class 'Automobile' in the ontology 'vehicles' within Ontolingua. From Farquhar, Fikes & Rice (1996).

knowledge engineers and within organisations. Section 2.6.2 outlines how these systems support collaboration in the construction and maintenance of ontologies.

2.6.1 ACCESSING ONTOLOGIES

There are two main purposes behind accessing remote ontologies: in order to build KBSs based on them and in order to learn about a domain. Ontology Servers, discussed in Section 2.6.1.1, fulfil the first purpose, while Corporate Memory Systems, discussed in Section 2.6.1.2, fulfil the second.

2.6.1.1 Ontology Servers

Ontology Servers provide ontologies for general use in knowledge intensive applications through the Internet. The ontologies provided by ontology servers are usually defined internally using a knowledge representation language such as KIF (within the Ontolingua server - Farquhar, Fikes & Rice, 1996; Rice et al., 1996; Farquhar et al., 1995: see Figure 2.8) or LOOM (in the Ontosaurus browser - Swartout et al., 1996: see Figure 2.7). This underlying knowledge representation is then translated into HTML (HyperText Mark-up Language: see Section 4.3.2.1) pages that can be viewed through the WWW (see Section 4.3.2). Farquhar, Fikes & Rice (1996) describe this process as having the knowledge 'projected through a variety of lenses'. The translation is

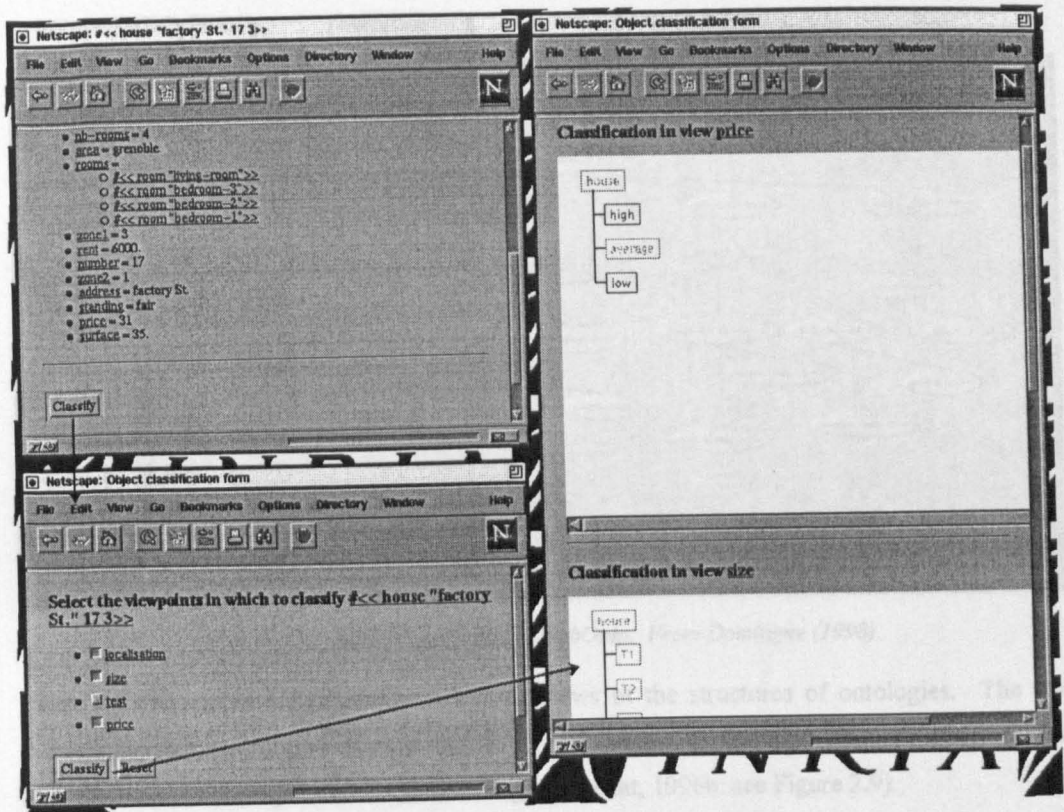


Figure 2.9: Viewpoints in Co₄. From Euzenat (1996a)

typically done through either CGI scripts², which interface between the knowledge representation and the HTTP (HyperText Transaction Protocol: see Section 4.3.2.1) server itself, or through a programmable HTTP server such as CL-HTTP³, the common-lisp HTTP server. The Ontolingua server also allows users to access the ontologies it holds either programmatically, through a networked API, or by having them translated into different knowledge representation languages, such as LOOM or CLIPS.

When accessed through the WWW, ontology servers generally offer a frame-based view of the knowledge represented within the ontology being viewed (see Figure 2.8 for an example from Ontolingua). Having a frame-based view of the ontology means that users view pages that represent either different real-world objects, such as 'Granite' (a type of rock), or conceptual classes which group several of them together, such as 'metamorphic rocks'. These pages give information about the object's location within a conceptual hierarchy (with links to other objects) and specific information about the object itself. This information can usually be altered through links or HTML forms within the pages, allowing the user to change the ontology as they

² More information about CGI is available at <http://hoohoo.ncsa.uiuc.edu/cgi/overview.html>

⁴ More information about CL-HTTP is available at <http://alpha-bits.ai.mit.edu/projects/iip/doc/cl-http/home-page.html>

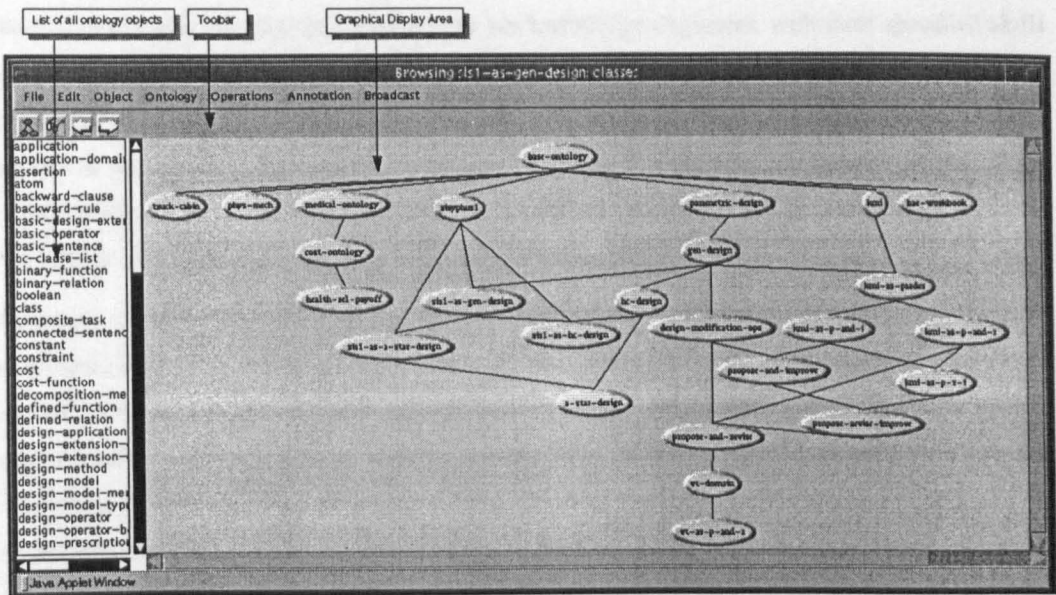


Figure 2.10: Editing ontologies in WebOnto. From Domingue (1998).

desire. Different servers offer other types of views of the structures of ontologies. The Co₄ system, for example, allows users to define sets of viewpoints that give alternative hierarchical classifications of the objects within the ontology (Euzenat, 1996b: see Figure 2.9).

More recent developments, like JOE (Java Ontology Editor: Mahalingam & Huhns, 1997), Tadzebao and WebOnto (Domingue, 1998: see Figure 2.10) use JavaTM applets as clients to ontology servers. The use of Java gives much more flexibility and interactivity in interface design than plain HTML, while maintaining much of the portability and platform independence of HTML.

2.6.1.2 Corporate Memory Systems

Corporate Memory Systems hold a great deal in common with Ontology Servers. As with ontology servers, they hold knowledge in some kind of semi-formal representation, with the aim of making it available for reuse. Under van Heijst, Schreiber & Wielinga's (1997) definition, Corporate Memory Systems serve *domain knowledge* rather than ontologies. The type of knowledge held by Corporate Memory Systems can be items such as telephone directories, internal reports, archived discussion, design rationales (see Section 3.5), customer databases, records of best practice and so on. These types of information might be made available on an organisation's Intranet (e.g. the TCE Corporate Technical Memory: Fisher, 1997) or through applications like Lotus Notes. Corporate Memory Systems aim to improve both the collection of and access to these resources.

⁴ More information about Java is available at <http://java.sun.com>

The users of Corporate Memory Systems are not knowledge engineers, with their specialist skills in understanding formal representations, but corporate workers. In addition, the primary purpose of the knowledge is not to create decision support systems but to inform physically and temporally separated workers, to support discussion between them, and to be added to. In other words, within organisations, knowledge management is concerned with promoting knowledge growth, knowledge communication and knowledge preservation (Steels, 1993).

Corporate Memory Systems need to support three processes: construction, distribution and use by employees (Dieng, Corby, Giboin & Ribière, 1998). While the initial construction of a corporate memory may be carried out by a knowledge engineer, corporate memories are designed for growth and evolution and thus need to support the addition of knowledge by people who are not knowledge engineers. This is often done either by automated document analysis (Trigano, 1994) or by incorporating the system into the normal work cycle: in other words, through incidental knowledge acquisition (see Section 2.5.2.3). The use of corporate memories depends on users being able to retrieve the information they want in the form that they want to view it. These issues will be discussed in Chapter 4, on Information Retrieval.

Research on corporate memory addresses the issues surrounding knowledge sharing from a computer supported cooperative work perspective. Ontology servers address the same issues from a knowledge engineering perspective. The constructivist viewpoint argues for a greater integration of the two areas, as it recognises the need for collaborative distributed ontological engineering.

2.6.2 COLLABORATION

Simply being able to access ontologies is not sufficient for collaborative ontological engineering. Ontological Engineers and even domain experts need to be able to discuss the ontologies and to contribute to them. Section 2.6.2.1 discusses how communication is supported in current ontology servers, while Section 2.6.2.2 examines the management of multi-user construction of single ontologies.

2.6.2.1 Communication in Ontology Servers

Ontology servers are designed to let a number of users create an ontology together, and thus need to support communication between them. Communication between users is particularly important for ontology servers designed to be used for corporate memory, so that the system becomes more than a simple record of events (Euzenat, 1996b). Examples of the types of communication currently supported are:

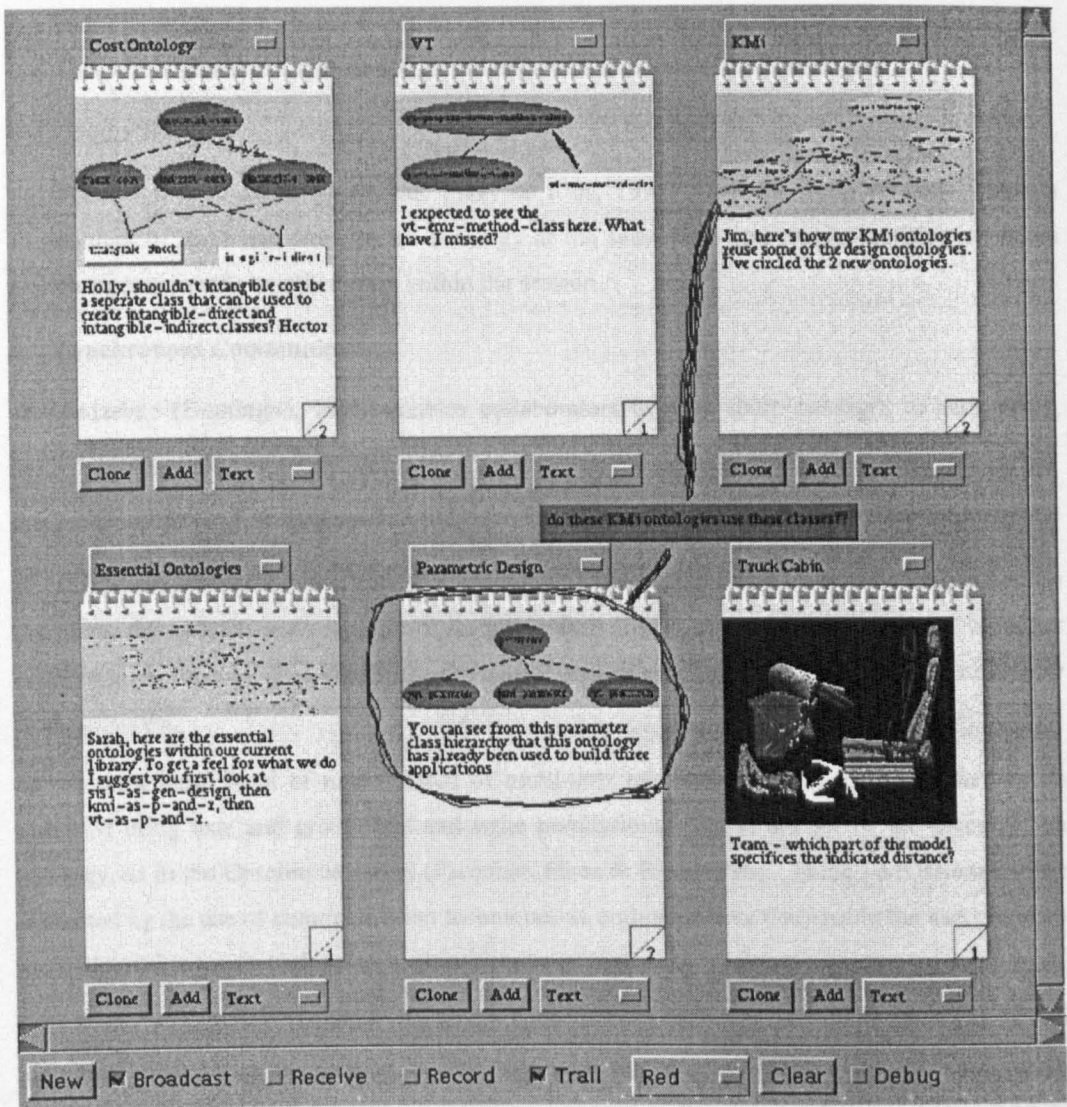


Figure 2.11: Exchanging messages in Tadzebao. From Domingue (1998).

- **Subscription**

Within some systems, such as SHADE (SHARed Dependency Engineering - Gruber, Tenenbaum & Weber, 1992), users can 'subscribe' to certain areas of interest within a knowledge representation. They are then notified of any changes that occur to those areas.

- **Annotations**

The support of human-readable ontologies often involves the use of basic annotations, both to make the axioms more easily understandable and to aid communication between multiple users who may be accessing the same ontology. HyTropes (Euzenat, 1996b), for example, allows users to add annotations to either the top or the bottom of any HTML page it generates. Skuce (1995) argues for a more formal annotation procedure, in which concepts are annotated

with the assumptions made for them (as outlined in Section 2.4) and examples of their use. This would generate a type of design rationale (see Section 3.5) for ontologies.

- **Group Sessions**

The Ontolingua server (Farquhar, Fikes & Rice, 1996) supports group sessions where a number of users can work on an ontology at the same time and receive notification when changes are made by other users within the session.

- **Synchronous Communication**

Tadzebao (Domingue, 1998) enables collaborators to send short messages to each other, including images and ontologies (see Figure 2.11).

2.6.2.2 Combining Ontologies

As well as supporting communication between users, ontology servers often aid collaboration by managing the changes users can make to ontologies. These changes could be based on either users' implicit conceptualisations of the domain or explicit ontologies they have created externally from the server.

At the most basic level of management of multi-user ontologies, access to ontologies can be restricted using user and group read and write permissions. These are set by the owner of an ontology, as in the Ontolingua server (Farquhar, Fikes & Rice, 1996). These permissions can be supported by the use of communication techniques as outlined above that enable the users to keep track of the changes that others are making.

Within the Ontosaurus Browser (Swartout et al., 1996), changes can only be made by an individual user if they are consistent with the rest of the ontology. This is possible due to the reasoning capability built-in to the LOOM representation language and prevents inconsistent ontologies from being built. This facility is not only useful for multi-user ontology construction, but also for checking the consistency of knowledge from a single user. However, as we shall see in Section 2.7, in some contexts, such as the representation of knowledge for human understanding, the representation of inconsistency is a benefit, if not a necessity.

Co₄ (Euzenat, 1996a & 1996b) goes one step further in structuring the interactions between individuals working on the same ontology. Ontologies are built into a hierarchy of group knowledge bases, each one of which represents the consensual knowledge for the users that are subscribed to that group. At the bottom of the hierarchy, each user can have a number of knowledge bases that they can alter as they please. When a change is proposed for the group base, each of the users subscribed to it is notified of the proposal and may either accept or reject it. If all users accept the proposal, the group knowledge base is changed, but if a single user objects, the change is not made in its present form. This procedure models itself on the submission procedure

for academic journals. Individual users can also choose not to accept any changes into their own individual knowledge bases, even if they are accepted on a group level.

In Swartout et al.'s (1996) vision of the future in knowledge base design through a collaborative ontology servers, knowledge engineers are able to familiarise themselves with the domain and construct an initial knowledge base from a pre-existing ontology. This ontology can be translated into the many formats, such as KIF, LOOM or C++, for use by the system they are building. If a knowledge engineer has knowledge to add to the ontology, they can 'check out' part of the ontology, update it, and then 'check it back in'. At this point, the system itself automatically assesses the new knowledge for compatibility with the previous version of the ontology. Other users are able to look at a record of the changes that have occurred to the knowledge base. This and other methods of combining knowledge from multiple experts will be discussed in Section 8.4.2.

2.7 KNOWLEDGE MODELLING

Knowledge Acquisition, as described in Section 2.5, results in a number of *knowledge level models* or *conceptual models* of a domain. These models can then be translated into a programming or knowledge representation language in order to produce KBSs. Uschold (1998) defines knowledge level models as:

Short Definition a MODEL constructed in a manner whereby no specific attention is paid to implementation issues and decisions. Typically, such a MODEL:

- expresses some portion of the KNOWLEDGE required by one or more agents to achieve an existing or required level of problem solving competence; or
- will be made during an early phase of KBS construction and serve to specify the requirements for subsequent design and implementation.

Knowledge level models have many uses for knowledge engineers. Uschold (1998) highlights the following:

1. System engineering benefits for building KBS
 - a. Specification: to assist the process of identifying requirements and defining a specification for a KBS
 - b. Knowledge acquisition: to drive model-based knowledge acquisition
 - c. Reliability: to assist evaluation of the correctness and/or completeness of knowledge

- d. Reusability: a knowledge level model can serve as the basis for more than one KBS, e.g. a formal ontology represents domain knowledge that may be (or may become through automatic translation) a reusable and/or shared component in a number of systems
2. To enhance human understanding and communication
 - a. To increase understanding of some area of interest, e.g. of how a human solves problems in a given domain
 - b. To document knowledge in an unambiguous manner, e.g. corporate knowledge assets

In knowledge level models for ontologies, the stress is on reusability (1d) and the documentation of knowledge (2b).

The other major dimension along which knowledge level models vary is in their *formality*. Knowledge is elicited from non-contrived knowledge acquisition techniques (see Section 2.5.1) as natural language transcripts, video recordings and so on: *informal knowledge level models*. Contrived techniques give more formal knowledge level models, such as repertory grids and classification hierarchies. These relatively informal models can be analysed by knowledge engineers to create even more formal models, usually using a knowledge level representation language such as CML (Schreiber et al., 1994) or Ontolingua (Gruber, 1993).

Buckingham Shum (1997) identifies three undesirable consequences of formal representations of information:

- They are less flexible than informal representations, preventing incomplete, inconsistent or ambiguous information from being represented.
- They are difficult to understand, which means that ontologies cannot be easily browsed by knowledge engineers or others learning about the domain in order to gain an understanding of a domain.
- It is harder to represent contextual information using them than it is using informal representations.

Uschold (1998), on the other hand, identifies three advantages of formal representations:

- Reduced ambiguity
- Capacity to automate analysis of various sorts (e.g. consistency, completeness, soundness)
- Capacity to automate subsequent KBS construction

The disparity between Buckingham Shum and Uschold on this point lies in the context in which they are imagining knowledge level models being used. Buckingham Shum focuses on knowledge use within organisations, as in Corporate Memory Systems (see Section 2.6.1.2), while

Uschold is mainly considering knowledge use for the construction of KBSs. For ontologies, which are designed for both purposes, there is thus a tension between fulfilling the system engineering requirements and the requirements for human understanding and communication.

Within an ontology server, the knowledge-level models presented to users can be different from the internal knowledge representation. For example, an ontology server need not hold a conceptual map as an image, but rather hold both the underlying knowledge that the conceptual map encodes, and an algorithm for converting that knowledge into a conceptual map. The knowledge content can be separated from the knowledge presentation (see Section 8.5.3). However, the internal knowledge representation used should combine the advantages of both informal and formal knowledge-level models, for example by both being able to represent inconsistent knowledge *and* check for consistency.

Knowledge representation systems are based around a number of ontological commitments that specify the types of concepts and relationships that are needed in order to represent knowledge. In their analysis of Frame Representation Systems (FRSs) for the construction of the Generic Frame Protocol (GFP: Karp, Myers & Gruber, 1995; Karp & Gruber, 1997), Karp, Myers & Gruber (1995) identified several ontological commitments common to FRSs. These are described here to introduce terms that are used when discussing knowledge representations throughout this thesis, particularly in Sections 5.4.2 and 8.3.1.

- **Frames** are named concepts within the domain. Classes, individuals, slots and facets are all frames. Frames can have slots and facets associated with them. (See Section 8.3.1.1.)
- **Classes** hold groups of frames as instances.
 - Classes are arranged in a hierarchy, such that *superclasses* of a class are those which hold a superset of the instances that it holds. Likewise, *subclasses* hold a subset of instances held by the class. The *direct superclasses* of a class are those superclasses which do not have any subclasses that are also superclasses of the class. Similarly, *direct subclasses* are those that do not have any superclasses which are also subclasses of the class.
 - Classes can fall into *partitions*, which are groups of classes in which none of the classes have instances that are instances of any of the other classes in the group (i.e. they are *exclusive*). If (some of the) subclasses of a class form a partition, it is known as a *subclass partition*; if a subclass partition covers all the instances of the class, it is termed *exhaustive*.
- **Individuals** are terms that are not classes. They are often real things within the domain, in the same way as entities in repertory grids (see Section 2.5.1.3). The classes an individual belongs to are known as its *types*.

- Slots hold values for frames. Values may be other frames or primitive values such as numbers or strings.
- The slots on a frame are determined by the classes to which the frame belongs. Slots that are defined on a frame (through inheritance from the classes to which it belongs) are known as *own slots*. They hold values specific to the frame itself. Slots that are defined on classes (and inherited by its instances) are known as *instance slots*. Instance slots hold the *default* values that are inherited by the class's instances. The classes on which a slot is defined gives the slot's *domain*.
- Slots have a *range* of values that they can take. The range of values can be restricted for certain frames.
- Slots can take a number of values for any one frame. The minimum number of values that can be taken is termed the *minimum cardinality* while the maximum number of values is the *maximum cardinality*. *Single value slots* take one value and one value only. The cardinality of a slot can be restricted for certain frames.
- Slots can have *inverses* where if a certain frame is taken as a value for another frame, on the inverse slot, that frame is taken as the value of the first.
- Slots can take the *same values* as each other. The slots that must take the same values can be restricted for certain frames.
- Slots can be *symmetric*, *asymmetric*, or *antisymmetric*, *reflexive*, *antireflexive*, or *irreflexive*, and *transitive*, *weakly transitive* or *intransitive*. These place logical restrictions on the values that can be taken for the slot.

It should be noted, however, that this analysis originated from a particular perspective and subscribes to the same ontological commitments as KIF and Ontolingua (Farquhar, Fikes & Rice, 1996). The translation between GFP/KIF/Ontolingua and other representation languages, like LOOM, is not a straight-forward operation (Uschold et al., 1998).

2.8 CONCLUSIONS

This chapter has described the area of knowledge engineering and outlined the relevant major issues in terms of collaboratively constructing an ontology.

The progress of the World Wide Web (WWW) is gradually bringing with it a change in the nature of documents. Instead of being static information that is simply transferred in the same way to whoever requests it, documents are becoming animated: growing and evolving information that adapts its presentation according to the needs of the user. Systems that support these *Living Documents* not only store and supply information, but support collaboration through structured

communication which itself becomes part of the document. In the same way, ontologies should develop into *Living Ontologies*, facilitating collaboration between their creators.

This chapter has mentioned some of the issues involved in constructing Living Ontologies and the requirements that would be made of a system that supported them.

- **Growth and Evolution**

Living Ontologies need to be able to grow and evolve. While knowledge engineers might be involved with the initial seed knowledge, this needs to be added to over time. Arguably, domain experts themselves should be able to add this knowledge, which brings with it the requirement for direct knowledge acquisition from domain experts (as discussed in Section 2.5.2.2). Alternatively, knowledge generated in the normal course of work should be automatically integrated into the ontology (as discussed in Section 2.5.2.3). These requirements lead to the following requirements for APECKS as described in Section 5.3.3.2:

- Users should be led step-by-step through knowledge acquisition process (requirement 9)
- The system should support contrived knowledge acquisition techniques for direct knowledge acquisition from users (requirement 10)

- **Flexible Internal Knowledge Representation**

As discussed in Section 2.7, Living Ontologies should have an internal knowledge representation that allows the translation of ontologies into forms usable in other knowledge-based applications. On the other hand, the degree of formality should be such that a certain degree of inconsistency, incompleteness and ambiguity is permitted, especially during a brainstorming stage of development. This leads to the following requirements in Section 5.3.3.2:

- Knowledge should be represented internally in a semi-formal knowledge representation (requirement 4)
- The internal knowledge representation should be transformable into a human-readable presentation (requirement 5)

- **Representations of Multiple Ontologies**

There is a need for a number of ontologies, generated on the basis of different sources on the same domain, to coexist. These coexistent ontologies can be viewed as examples from the design space for that particular domain. The comparison of these ontologies can lead to a greater understanding of the design space itself by making explicit the assumptions on which they were created. They can also inform the integration of ontologies with each other. This leads to requirements in Section 5.3.3.2 and Section 5.3.3.3:

- Roles should be considered separately but retain a link to other roles within the same domain (requirement 8)
- Knowledge representations should be compared in an automated fashion (requirement 11)
- Prompts should be generated for users to make changes to and discuss roles on the basis of comparisons (requirement 12)
- **Support for Communication**

Constructing Living Ontologies is a collaborative and distributed enterprise. There thus needs to be support for many forms of communication, both structured and unstructured, and formal and informal. Formal communication, in the form of rationales for the construction of ontologies, should be built up so that the decisions taken during their construction can be reviewed, and changed or repeated as necessary in future versions. The requirement for communication support in systems supporting collaborative work is discussed more thoroughly in Chapter 3.

CHAPTER 3 DISTRIBUTED COMPUTER-MEDIATED COMMUNICATION

3.1 SUMMARY

This chapter examines the issues surrounding the support of collaborative work through computer-mediated communication. As organisations are becoming more distributed and more people work from home, there is more demand for technologies that can support collaboration over a distance. The increase of computer power and network bandwidth can support the development of computer-based tools which do just that.

One of the most important aspects of collaborative work is communication with other people. In this chapter, I first describe some of the issues concerning the design of Computer-Mediated Communication (CMC) systems. I then focus on two technologies that support collaboration in different ways, focusing on the communication aspects of the technologies. These two technologies are:

- Text-based Collaborative Virtual Environments; and
- Design Rationale systems

The discussion of these technologies lead to the identification of a number of requirements for CMC applications:

- **Accessible:** users must be able to access the system with their current technology.
- **Learnable:** users must be able to learn how to use the system within a short period.
- **Persistent:** users should keep the same identity across many sessions.
- **Archivable:** users must be able to reread or replay previous communication.
- **Authorable:** the system must be easily initialised with information to be made available to users.
- **Programmable:** the system should enable organisers to automate appropriate tasks.
- **Reusable:** the system should be reusable in many situations.

The archiving of communication within such systems leads us to another important issue in computer-supported collaborative work, namely information retrieval, and this is discussed in Chapter 4.

3.2 INTRODUCTION

Communication is a vital part of all types of work, be it in instruction, negotiation or socialisation. For collaborative work, in which people work together to achieve a common goal, communication is the central aspect. Buckingham Shum (1997) also argues that communication is the central process in 'knowledge work' (Kidd, 1994). Enhancing communication therefore improves performance in the workplace.

There is a vicious (or beneficial, depending on the outlook taken) cycle linking enhancements in communication technology to the growth of distributed work. As distributed work becomes more popular, the demand for new, effective, communication and collaboration technologies grows. These new communication technologies allow organisations to become more dispersed, increasing distributed work, and so the cycle begins again.

Technology enables communication to occur where it otherwise would not, and it reduces the cost of communication by negating the need for travel. In addition, there is the prospect that computer-based communication can actually enhance collaboration above that of normal, face-to-face communication. Of course, technology is not the panacea for all communication needs, and its effectiveness depends on both the technology itself and the organisational and social context in which it is placed.

The central motivation of this thesis is to help people work together more effectively through the sharing of knowledge. As such, investigating communication methodologies and technologies is very important. In this chapter, I first give an overview of the types of computer-mediated communication technologies that are available, and then focus on two such technologies: text-based Collaborative Virtual Environments (CVEs) and Design Rationale, or Group Decision Support systems. These technologies offer complementary approaches to computer-mediated communication, both of which highlight requirements for Computer-Supported Co-operative Work (CSCW) systems. In addition, as we shall see in Section 5.2, the central task supported by the system described by this thesis (APECKS) is one of *design*. Support for design is the central purpose of Design Rationale systems, and the methodologies they use are therefore informative to the design of APECKS.

3.3 COMPUTER-MEDIATED COMMUNICATION

Computer-Mediated Communication (CMC) systems aim to support communication between two or more people using a computer as a mediator. Such systems have been found to be less effective than face-to-face communication in terms of socialisation (Markus, 1994), task outcomes (McCarthy & Monk, 1994) and user satisfaction (Olaniran, 1996). This is due to factors such as the decreased expressiveness of the medium, the time delays involved in passing messages and the lack of social cues present in the communication. CMC research seeks to demonstrate the

feasibility of new systems that, it is hoped, alleviate the difficulties inherent in using new communication methods. What is more, new CMC systems aim to enhance collaboration over and above that which is possible in normal, face-to-face, communication by taking advantage of their benefits in, for example, encouraging more equality in communication (Kollock & Smith, 1996).

Fischer et al. (1992) make the distinction between *direct communication* and *indirect communication*. Direct communication is simple person-to-person communication, talking being the most obvious example. Direct communication is not necessarily instantaneous or sequential: a letter is an example of direct communication, and an unanswered letter still communicates information to the receiver. Indirect communication occurs through an intermediary. This might, for example, be a solicitor or chairman. Unlike direct communication, indirect communication requires the active participation of both the provider and the receiver. In order to receive information, the receiver has to make contact with the intermediary.

Advances in CMC systems have altered both direct and indirect communication. Direct communication technologies range from telephones and tele-conferencing, through email and video-conferencing to Collaborative Virtual Environments (see Section 3.3.1). In these systems, although messages pass through the computer, it acts as a channel rather than an intermediary: the receiver of the message does not have to request it. Computers are used as intermediaries in CMC systems that support indirect communication, however. Design Rationale systems (see Section 3.5) are an example of how indirect communication can be used to allow people to keep in contact with each other. Each individual member of a group can supply information to the argumentation, and other members can read this information when they require it.

One advantage that indirect communication holds over direct communication is that it can be incorporated more easily into the work of the individual. This means that supplying information is not done at such a cost (Fischer et al., 1992). The more the supply of information is incorporated into the normal work of the members of the group, the easier indirect communication becomes. In this way it is similar to incidental knowledge acquisition (see Section 2.5.2.3), and, indeed, indirect communication can be used as a KA resource.

With both types of communication, the provider of information has to supply information that is useful to the other members of the group. Often, the information supplied will not be useful at that particular moment, or to many of the members of the group. These are problems of *timing* and *appropriateness*. Indirect communication aids in the problem of the timing of information, since information can be held indefinitely, but furthers the problem of appropriateness since information seekers will not necessarily know where to look for the information. Issues involving the retrieval of information from such systems are discussed in more depth in Chapter 4.

In this section, I first give an overview of one of the important subsets of CMC applications, Collaborative Virtual Environments, on which I will be focusing. I then go on to describe a framework for describing CMC applications, which indicates some of the important features that should be considered in their design.

3.3.1 COLLABORATIVE VIRTUAL ENVIRONMENTS

Collaborative Virtual Environments (CVEs) are a particular form of CMC system that embed communication within a virtual environment. They are of particular interest because they give a context in which communication can take place, enabling the use, to a varying extent, of non-verbal cues which can alleviate some of the problems of computer-mediated communication.

CVEs are often categorised as either *graphical* or *text-based*. At one extreme, graphical CVEs are characterised as rich three-dimensional graphical worlds that are viewed through headsets and with which users directly interact using data-gloves or even more immersive technologies. Communication between users is similar to real-life communication, using audio and augmented with non-verbal communication through video. At the other extreme, text-based CVEs are characterised as worlds described with text, with which users interact using a command-line interface and in which text is the only means of communication.

In fact, graphical and text-based CVEs lie on a continuum that includes, for example, primarily text-based CVEs that use graphics instead of textual descriptions of rooms and primarily graphical CVEs that use text-based communication. All types of CVEs have the following in common:

- **Spatial Representation**

A virtual environment involves the representation of a space with which users can interact. It is possible for virtual environments to be representations of any conceivable space. However, they typically mirror the real world both in terms of their spatial characteristics (such as being three-dimensional and having Euclidean geometry) and in their content (buildings with rooms containing filing cabinets, for example). Typically, the content of a virtual space reflects its purpose: a virtual environment for collaborative learning, for example, could contain a virtual university building, with classrooms and lockers. Abiding to spatial characteristics and content found in the real world makes the environment more familiar to new users. However, there are situations where it is less appropriate, such as allowing portals from one side of an environment to another; making a virtual representation of the inside of a human body in order to teach anatomy; or representing the results of a database search within a virtual world as in VIBE (Olsen et al., 1993: see Section 4.4.1).

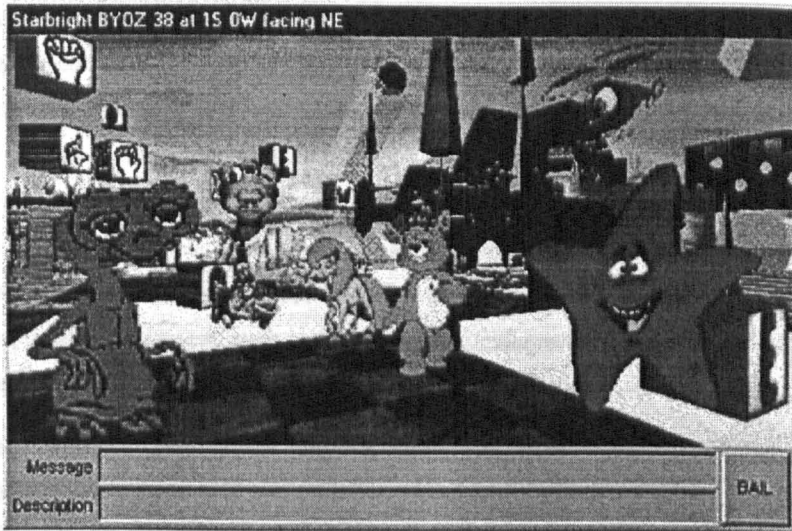


Figure 3.1: Screenshot from *Starbright*¹, a virtual chat environment for hospitalised children that uses *Worlds Chat*.

- **Interaction**

The interaction between users and the virtual environment almost always involves the ability to move through the environment, and often the ability to interact with objects within it, such as by sitting on chairs. In CVEs, often all the objects the user can interact with are tools designed to aid collaboration, such as whiteboards (for making notes or drawing on) or filing cabinets (for storing information).

- **Immersion**

Immersion within a virtual environment is the sense of 'being there' experienced by the user. The degree of immersion users have within a CVE affects the ease with which they can interact with the environment and with other users, and therefore the effectiveness of the collaboration. Cognitive immersion within a CVE is distinct from, but probably influenced by, the amount of sensory immersion given to the users: the amount of synthetic stimulation of users' senses (primarily sight, sound and touch) given by the CVE (Hand, 1996). Cognitive immersion may also depend on the modes of interaction the user has with the environment, such as whether it is a command-line interface or a graphical user interface.

Internet-accessible multi-user graphical Virtual Reality (VR) systems that can be run on standard machines, such as *Worlds Chat*⁵ (see Figure 3.1) and *CYBERSpace STUDIOS' 3D Virtual Worlds*⁶ are becoming more common, but these rely on (often fairly restricted) text-based communication.

⁵ Information about *Worlds Chat* can be found at <http://www.worlds.net/>

⁶ Information about *CYBERSpace STUDIOS' 3D Virtual Worlds* can be found at <http://www.lyrastudios.com/cyberspace/>

Systems such as MASSIVE (Greenhalgh & Benford, 1995), which support real-time audio communication can currently support only 10 mutually aware users but it is predicted that VR systems based on multicast, such as MASSIVE-2 (Benford, Greenhalgh & Lloyd, 1997), should be able to support in the order of 100 mutually aware users. However, bandwidth constraints mean that real-time audio communication over the Internet is, at present, unmanageable when users are separated by continents, so these systems currently often fall back on text communication between users. In addition, for more sophisticated interfaces, more computer power and specialist equipment, such as headsets, is advantageous, but this high-end equipment is not commonly available at the present time.

In text-based collaborative virtual environments, the environment itself, as well as the communication between users, is given in text. At present, text-based CVEs give a good payoff between richness of the CVE, in terms of both the presentation of the environment and expressiveness of communications between users, and usage of computer resources. While text-based CVEs do not provide a sophisticated user interface, they are still arguably the most common and most used types of CVE, although this is becoming less true as bandwidth and computer power increase. Text-based CVEs are not limited in their spatial representations: it is possible to build four-dimensional and non-Euclidean worlds. In addition, complex actions, such as moving to 'The Library', can be easily expressed with text-based commands (`go library`): attempts have been made to integrate these types of commands into graphical virtual environments (Bersot et al., 1996). Studies have shown that cognitive immersion is not reliant on sensory immersion (Tromp, 1995), and low-bandwidth, text-based CVEs can provide a sense of 'being there' without direct sensory input. Text-based CVEs are discussed in detail in Section 3.4.

3.3.2 DESCRIBING CMC SYSTEMS

In CMC research, the computer application used can be viewed as the medium in which people communicate. Different media are best suited to different tasks, and thus the choice of medium should reflect the qualities of the task which it is designed to aid (Kraut et al., 1992; Reid et al., 1996). The classification of CMC systems is important, therefore, since it helps us identify those features which alter how effectively they are used for different types of tasks. By integrating studies of CMC systems, a descriptive framework that can be used for such a classification has been constructed and is presented here. An outline of the framework is given in Figure 3.2.

3.3.2.1 Channels

The first concept that aids in the classification of Collaborative Virtual Environments (CVEs) is that of *channels of interaction* (McCarthy et al., 1993). Each channel uses a different medium to pass messages to or from the user. Channels that take information from the system or other users to the user can be termed *output channels* and those that take information from the user to the

Output Channels:	How many channels are used to present data to the user?
Input Channels:	How many channels does the user use to send data to the system?
Correspondence:	How do the input and output channels correspond to each other?
For each channel:	
Medium:	What medium is used by each channel? (text, hypertext, images, sounds)
Functions:	What are the purposes of the data the channel passes?
For each function:	
Granularity:	How large are the messages that are carried by the channel?
Interactivity:	How quickly are messages passed between those using the channel?
Descriptors:	What information about the source of the message is given?
Structure:	How much is the structure of the message enforced?
Permanency:	How long is the message retained within the system?
Access:	Who has access to the record of the messages passed within the system?

Figure 3.2: Framework for describing CMC applications.

system or other users, *input channels*. For example, drawing on a shared whiteboard involves an input channel that uses mouse-actions and an output channel that uses graphics.

Medium

The *medium* used by the channel is important. *Hot media* present information literally whereas *cold media* only describe the information (Curtis, 1992): for example, a sound could be played through audio media, but could only be described textually. Media can also differ in their inherent *expressiveness* (Kraut et al., 1992) in that some are better at transmitting nuance and subtleties than others. For example, the audio medium, when used for communication, transmits nuance through tone of voice, whereas text-based communication cannot do so.

Correspondence

The *correspondence* between the channels used for input and output is important: some VR systems, such as MASSIVE (Greenhalgh & Benford, 1995) allow users to input typed text while other users perceive this communication as sound. More than two input and output channels can also be tied together: in real life, we use both speech and visual signals to communicate with each other, and the correspondence between verbal and non-verbal utterances communicates something in itself. In this way, the combination of input and output channels also adds to the expressiveness of the system for communication.

3.3.2.2 Functions

Each channel can be used for many different *functions*, including interacting with a virtual environment as well as communicating with other users. It is important to consider these functions separately in order to differentiate between systems that use the same sensory channels for different purposes. For example, systems with an audio channel could use it for synchronous communication between users of the system, or it could simply be used for atmospheric sound effects.

Each of these functions can then be described both in terms of a *message-passing system* and a *shared data structure* as suggested by McCarthy et al. (1993). For example, email involves the passing of messages between two (or more) people, and it can also be thought of as the building of a library of messages that the user has received.

Message-passing

Functional channels can be described in terms of the *granularity* of the messages and the *descriptors* that are attached to the messages. The granularity of the message ranges from a continuous stream of information, such as audio communication, to long utterances, such as letters or WWW pages. Small-grain messages also tend to lead to higher *interactivity* (Kraut et al., 1992) using the channel, the speed at which turns take place, although this is not always the case. What is commonly known as *synchronous* communication is simply message-passing with high interactivity. Descriptors placed on the messages give meta-information about the content of the message, such as the author of the message; when it was sent; keywords; who it was intended for; and who has read it.

Communication can be *structured* both in terms of the information that is required to be passed within each message and in terms of the types of messages that can be used to respond to one that has been received. In systems based around Speech Act Theory, the purpose of each message is specified by the participants and they are prompted to follow certain sequences of utterances. For example, in The Coordinator (Winograd, 1988), the system prompted its users to send certain types of emails to each other in response to the types sent to them. Similarly, Design Rationale systems as described in Section 3.5 limit the type and format of the messages that can be generated. Structured communication in which there is no intermediary requires that the participants understand explicit or implicit rules governing their exchanges, such as in word games or wedding ceremonies.

Shared data structure

When a CVE is considered in terms of a shared data structure, the major classifications that can be made are a) the *permanency* of the data and b) the *access* that the users of the system have to the data (McCarthy et al., 1993). Virtual environments are particularly easy to view as data stores

since they are usually constructed from an external representation (e.g. a database) which can be changed by users through their interaction with the environment. For example, picking up an object in a VE and moving it somewhere else causes a permanent change in the VE. The permanency of communication between users is interesting as it has been suggested that the very impermanence of utterances, as in normal speech, may lead to more efficient communication between participants (McCarthy & Monk, 1994). On the other hand, permanency is necessary for both indirect communication and the archival of information (see Section 3.4.2.3). The access levels of different users are important as they determine who can communicate with whom and the degree to which users can manipulate the environment.

3.4 MULTI-USER DOMAINS

This section describes Multi-User Domains (MUDs) and, more specifically, MOOs (MUD - Object Oriented: Curtis & Nichols, 1993) as examples of text-based Collaborative Virtual Environments⁷. APECKS was constructed on top of a MOO (see Section 5.4.1) and thus incorporates the capabilities described here. MUDs use conversational or command-line interfaces and present users with textual representations of an environment using paragraph-long descriptions of rooms. They have a client-server architecture whereby a central server computer holds the program running the MUD and users connect to it from local computers using client software. An example extract from an interaction within a MOO is given in Figure 3.3.

To give an idea of the usage of MOOs, there are currently around 20 large public MOOs world-wide (Fox, 1997), along with around the same number of more restricted MOOs (Schneider, 1997) and a large number of small, start-up MOOs. As an example, MOOtiny, which was a relatively small public MOO hosted at the University of Nottingham, had approximately 600 users and over 1000 rooms, of which 700 had at least one exit or entrance and were therefore connected with the rest of the environment. In the largest public MOO, LambdaMOO⁸, it is not unknown for over 100 users to be connected simultaneously, though most MOOs have a maximum attendance of around 20 to 30 users at any one time.

MUDs can be compared to other online communications tools such as Unix talk and Internet Relay Chat (IRC) channels, in that they allow real-time text-based communication (Evans, 1993). Like IRC channels, this communication can continue between many people at the same time, rather than being limited to dyads as Unix talk is. MUDs are, however, CVEs, and thus also contain representations of a virtual space, with rooms and objects within them. This means that

⁷ Within this thesis, MUD is used as a general term for multi-user text-based virtual environments, MOO as the specific application.

⁸ LambdaMOO is accessible at lambda.parc.xerox.com, port 8888.

```

1  You have new mail (1) from Florence (#123)
2  >@next
3  Message 1 on Rose (#33):
4  Date:      Wed Sep  9 12:39:27 1998 BST
5  From:      Florence (#123)
6  To:        Rose (#33)
7  Subject:   Meeting
8
9  When you arrive, come and join us at the Cafe.
10 -----
11 >'Florence I've just arrived... On my way.
12 Your message has been received.
13 >look
14 The beach's soft white sand is cool under your feet, but the evening air is
15 hot.
16 Exits: cafe
17 >cafe
18 The cafe is cool after the humid heat outside, the air pushed gently around
19 by large ceiling fans. From the deck, you can see the red glow of the
20 setting sun reflected in the clear sea.
21 Florence and Jake are sitting at a table.
22 Exits: beach (out)
23 >wave
24 You wave.
25 Jake exclaims, "Hi!"
26 Florence smiles, "Hello :)"
27 >sit table
28 You sit down at a table.
29 >'Florence How are you?
30 You ask Florence, "How are you?"
31 Jake sighs.
32 >grin Jake
33 You grin at Jake.
34 Florence says, "I'm OK."
35 Jake orders a beer from the bar.
36 >:tries to catch the eye of the bartender.
37 Rose tries to catch the eye of the bartender.
38 The barmaid brings Jake a beer.
39 Jake says, "Type: order <drink>"
40 >order cider
41 You order a cider from the bar.
42 The barman brings you a cider.
43 Jake drinks some beer.

```

Figure 3.3: Example interaction within a MOO.

3.4.1.3 Functions

users of a MUD can not only communicate with each other but also interact with a persistent but ever-changing environment.

Within MUDs, all information about the environment is given to the user in the form of text, and the user interacts with the environment using a command-line interface. Users act on the environment through natural-language-like commands, where the creator of the object on which the user is acting determines the effect of the command. Because the feedback for actions is given textually, this leads to an interesting dichotomy between *programmed actions* and *uttered actions* (Carlstrom, 1992). Programmed actions are usually 'realistic' and have consequences (see lines 40-42 in Figure 3.3), while uttered actions are purely conjured through a user 'emoting' (see lines 36-37 in Figure 3.3). The use of emoting means that users can appear to carry out impossible actions, such as walking on the ceiling, that become 'real' to other users simply through their utterance (Cherny, 1995).

3.4.1 CLASSIFICATION

Below, I discuss the characteristics of MOOs in terms of the classification framework outlined in Section 3.3.2 above. An examination of the advantages and disadvantages of using MOOs for collaboration in more general terms follows this.

3.4.1.1 Channels

The only output channel supported by MUDs is text and, while it varies between MUD clients, the only input mechanism typically used is typing through a command line. Some MUD clients also provide iconic buttons that enable the user to carry out common actions. The input and output channels correspond to each other in that commands entered textually by a user cause textual outcomes.

At first, when considering the medium of text for communication, it seems to lack expressiveness compared to face-to-face communication. However, as Reid et al. (1996) point out, text can be expressive when the writer is given time, as shown in many poems and novels. Even without time, text can be used to transmit nuance through the use of emphasis (e.g. **asterisks** or *_underscores_*) and smileys (e.g. *:)*, *;)* or *: (*) as seen in line 26 in Figure 3.3. Equally, back channels, which give non-verbal feedback to the speaker, are possible in MOOs through the *emote* command (used by Jake in line 31 of Figure 3.3) and various common ‘feature objects’ (used in lines 23-24 and 32-33 of Figure 3.3) which enable users to smile, wave, shrug and so on (Cherny, 1995). However, both these means of adding expressiveness to MOO communication require effort on the part of the user: It is impossible to ‘betray meaning’ through body language on a MOO.

3.4.1.2 Functions

The richness of MUDs comes from the variety of functions that they support. In MOOs, communication between users can occur through normal speech (lines 25-26, 29-30, 34 and 39, of Figure 3.3), private paging (lines 11-12 of Figure 3.3), the use of *emotes* (line 31 of Figure 3.3) and private *emotes*, shouting, and through the internal mail system (lines 1-10 of Figure 3.3). Users of MOOs also use the command line to interact with the virtual environment, both to get more information about objects (through looking at them) and to perform actions with them (such as taking them). At higher levels, MOOs also allow the users to change the environment, through altering the descriptions of objects or even programming new actions that can be taken with them.

Speaking and emoting both involve passing small to medium sized messages between users. Typically, users type a sentence that is sent when the return key is pressed. The recipients of the message usually see the message immediately upon its being sent to the MOO (though this depends on both intra-MOO and Internet delays), which means the messages are reasonably synchronous. However, because of the time delay involved in classical turn taking using text-

based communication, users often overlap two or more threads of conversation that are carried out at the same time (McCarthy et al., 1993).

Speaking and emoting both involve the same message descriptors being used: the name of the speaker is added at the beginning of the message. Some MOO systems enable users to direct speech to individuals (e.g. lines 29-30 of Figure 3.3), and all private speaking and emoting have the recipient of the message indicated. Normal speech and emoting is restricted to the virtual room in which the user is located: all other active users in that room can hear what others say or do. Private speech and emoting is shared only between two users and shouting involves all the connected users receiving the same message. There is no means to go back and edit comments that have been previously made, and there is usually no archiving of synchronous conversations within the MOO. Individual users can, however, (depending on the client they are using) record conversations they participate in and save them for future reading. Indeed, even using the simplest telnet interface, users always have before them the last few minutes of conversation and can usually scroll up to see more.

The internal MOO mail system allows users to pass messages to each other asynchronously (see lines 1-10 of Figure 3.3). These messages have, like email messages, medium to large grainsizes and are not highly interactive. Also like email messages, descriptors such as a subject line, recipient and time sent as well as the name of the author are added to the message. MOO mail can be sent to individual participants, in which case it can only be read and edited by the recipient, or to mailing lists that have variable read, write and edit access. Both the mailing lists and the private mail of users are stored indefinitely within the MOO and can be reread by those with access.

Interaction with the environment typically involves messages of small to medium grainsize passing between the system and the user in a highly interactive manner. For example, in Figure 3.3, lines 40-42, typing 'order cider' immediately generates the responses 'You order a cider from the bar.' and 'The barman brings you a cider.' (providing that there is a bar serving cider in the room). System messages about the environment do not attach any descriptors that indicate that the message is from the MOO itself rather than a player. Indeed, users can pretend to be the system and send messages about the environment (known as spoofing, and usually considered a breach of etiquette, Curtis, 1992), and equally the system can pretend to be a user. (MOOs are ideal environments for Turing tests: the user can never know whether the individual they are interacting with is a true person or a software agent, except through the content of the conversation.) Messages from the system to an individual are private to that individual, although when the actions undertaken by an individual affect the environment, those changes are readily perceivable by other users, as in lines 35 and 38 in Figure 3.3. Actions that affect the

environment are permanent, in that they change the database on which the MOO is based, so that after Jake is brought some beer in line 38 of Figure 3.3, he can drink it (line 43).

The building and programming of MOOs is the most abstract level at which users can interact with the system. The message size for these interactions tends to be medium or large and, although the system immediately confirms whether changes have been made or not, the effect of the changes (such as whether a function works or not) is not readily apparent. Programming and building are entirely private affairs and can only be done by users with the right permissions. Changes that are made to the virtual environment in this way are persistent.

3.4.2 ADVANTAGES AND DISADVANTAGES

This section highlights the requirements made on CMC systems for supporting collaborative work over and above simply allowing multi-user communication. It also discusses the extent to which MOOs address these requirements, and thus the advantages and disadvantages of using them compared to other CMC systems.

3.4.2.1 Accessibility

Simply gaining access to a CMC system is one of the major barriers standing in the way of their use (Olaniran, 1996). MOOs are accessible through the Internet to users with relatively cheap technology and, compared to graphical CVEs, use relatively low bandwidth. Up until fairly recently, graphical CVEs required high computer power, good network connections and specialised equipment that was not currently widely available outside graphical VR research groups. MOOs, on the other hand, require only a telnet interface to give access to a populated CVE.

3.4.2.2 Learnability

CMC applications need to be usable by workers with the minimum of specialist training (Olaniran, 1996). There are two parts to learning how to use traditional MOOs. The first part is learning how to use a telnet or MOO/MUD client: this is heavily dependent on the application used to access the MOO. The second part is learning how to manipulate the virtual environment and communicate with other people through the command language used within the MOO. Both these form barriers for prospective users of the system.

A second learnability problem relates to people having less motivation to use MOOs than they do many other Internet modes of communication. Firstly, MOOs utilise a text-based interface that does not look very appealing, and secondly, they have adventure-game connotations that imply informal usage. On both these counts, MOOs are unattractive as serious tools for collaborative work. In order to persuade organisations to invest time in training their employees to use

specialist clients and a command language, it has to both capture their attention and be seen as a worthwhile pursuit.

Virtual environments such as MOOs do have an advantage over other CMC applications, though, in that they can be used to organise information into a metaphorical environment. In the same way that computer files and applications can be arranged through the Desktop metaphor, manuals, memos, minutes and other organisational information can be organised within a workplace metaphor. This should, in theory at least, enable users to locate material quickly by relying on their preconceptions about where such information would be located, thus making the system more usable (Smith & Wilson, 1993). The advantage of using spatial metaphors in presenting information to users is discussed in more depth in Section 4.4.2.

3.4.2.3 Archivability

One of the great benefits of computer-mediated communication is that it can be recorded automatically and revisited later. Activity within MOOs is not generally archived, but they are very easy to extend such that part or all of the activity within them can be stored for future use. For example, in MOOs used for electronic conferences, sessions are recorded so that those who cannot attend or who are in a different time-zone can still gain benefit from them (Hardy et al., 1997a; Hardy et al., 1997b; Hardy et al., 1998). The management of communication archives and the retrieval of information from them then become problems. These are discussed in Chapter 4.

3.4.2.4 Persistency

Individual users of a CMC system should be identifiable, to provide both social identity cues and greater group cohesion. These both may lead to better communication outcomes such as increased consensus formation and group-oriented communication (Sproull & Kiesler, 1986; Reid et al., 1996, Kollock, 1996). Users of MOOs are each represented within the virtual environment by an 'avatar'. These avatars can be customised by the user, and are persistent throughout the life of the MOO. Users can change the name and description of the avatar, and other users are not permitted to take any name currently being used.

3.4.2.5 Authorability

In order to take advantage of the context that virtual environments can give to communication, CVEs must be able to adapt to the varying contexts that are required of them. For example, they should represent auditoria for lectures, meeting rooms for small discussions, and a relaxing environment for social communication. Managers of CVEs need to be able to easily create or edit environments as required. MOOs use only text to describe the virtual environment they represent. Therefore, those running the environment do not have to learn how to use 3D-design software to create buildings, rooms or avatars, and indeed ordinary users can create the kinds of environments that they require. In addition, MOOs are quite well developed, with many of the tools that are


```
#170:categorical_slots this none this
1: "Syntax: $class:categorical_slots() => LIST {slot, class}*";
2: "";
3: "Returns a list of the slots within the role which are categorical: that
is, each frame can only take one of a certain number of discreet values for the
slot. If the slot has a slot domain of a single class, or a number of classes
which share a common ancestor, the subclass partition is defined on that class.
These slots can be easily changed into subclass partitions.";
4: slots = {};
5: for s in ($set_utils:union(this.instance_slots,
@$list_utils:map_prop(this.subclasses, "instance_slots")))
6:   if ((cats = s.range[2]) && !(s.range[1] || s.range[3] || s.range[4] ||
s.range[5] || s.range[6]))
7:     if (length(class = s.slot_domain) <= 1 || (class =
$set_utils:union($list_utils:map_prop(s.slot_domain, "superclasses"))))
8:       if (!class && s.default_cardinality == {1, 1})
9:         slots = (@slots, {s, this.role});
10:      else
11:        card = s:instance_cardinality(class = class[1]);
12:        if (card == {1, 1} || card == {0, 1})
13:          slots = (@slots, {s, class});
14:        endif
15:      endif
16:    endif
17:  endif
18: endfor
19: return slots;
20: "Last modified Wed Jun 24 11:37:33 1998 BST by Miss_X., #154@APECKS
Development MOO.";
```

Figure 3.4: Example MOO code.

useful for collaborative work, such as an internal mail system and various specialist rooms, not to mention the basic building blocks of rooms and avatars, available to organisations.

3.4.2.6 Programmability

Many of the functions required of CMC systems involve a degree of automation. MOOs are programming environments as well as virtual environments and are constantly being extended by their users. The MOO language (Curtis, 1996) is a rich one, based on C++ with error-handling similar to Java™, and MOOs can be programmed to automatically carry out any number of tasks. Figure 3.4 gives an example of a MOO 'verb'.

3.4.2.7 Reusability

Once time and money has been invested in creating a virtual environment for a particular situation, the results of that investment should be available to the rest of the organisation. MOOs are based on object-oriented databases, which means that individual objects or collections of objects can be reused within separate MOOs as necessary. Also, 'generic' objects can be built which exhibit or allow certain behaviours in all their descendants. Thus, tools developed for a particular purpose can be reused in other situations.

3.4.3 WWW-ENHANCEMENT

The major disadvantage of MOOs is the interface that must be used to communicate with other users and interact with the environment. There are two ways that the interface can be improved.

The first is by using a specialist MOO client with a more usable interface than most telnet clients, such as tkMOO-light (Wilson, 1994) or MacMOOSE (Bruckman et al., 1996), which support whiteboards as well as other useful features for supporting collaboration. The second is to add a different protocol for accessing the MOO, such as HTTP (Hypertext Transaction Protocol, the protocol used on the WWW: see Section 4.3.2.1), and thus allow people to connect to it using clients they already use regularly, in this case web browsers. The former has the advantage of an interface specifically designed for accessing MOOs, while the latter does not involve the users having to learn an entirely new application, as they can simply use their standard web browser.

Enabling the access of MOOs through the WWW transforms text-based CVEs into hypertext CVEs. This combination of technologies has been pursued by a number of MOOs (Fox, 1997). Accessing MOOs through the WWW changes slightly the characteristics of MOOs that were outlined in Section 3.4.1.

Firstly, the WWW enables the use of *hypermedia* such as images or sounds, as well as plain text, in describing the environment. This means that the medium is more expressive than text-based CVEs, although still not as rich as graphical VR systems or, indeed, real life.

Secondly, while communication between users themselves is largely unaffected, the grainsize of the messages exchanged between users and the VE are much larger. Each message is the size of a web page, and only the user can initiate message passing (through following a link or submitting a form). Indeed, the fact that users have to request messages from the system, and the delay involved in receiving them, leads to the large size of the messages: as much information as possible needs to be transmitted by the system when the opportunity presents itself. These messages also contain a lot more message descriptors, since the protocol used to transmit web pages, HTTP, involves sending meta-information about the content of the message, although this meta-information is not normally viewable by the users.

The primary advantage of enhancing MOOs with WWW technology is that graphical WWW browsers provide a direct-manipulation interface to the MOO (Schneiderman, 1982 & 1983). This means that users do not have to learn a set of MOO commands in order to interact with the environment or each other. Simple and common commands can be replaced by clickable links on a web page. For example, a WWW page that represents a room within the MOO can have links for each of the exits from the room: instead of typing the direction of the exit, the user need only click on it. However, not all commands are simple enough to be enabled through a single click. Because of this, image maps and HTML forms have been added as means of interacting with the objects within the virtual environment.

Several MOOs have started using Java applets to provide access through the WWW. Most of these Java applets offer the same functionality as specialised MOO clients, with a command-line interface and textual feedback, and are thus not an improvement over normal MOO clients.

However, they can be used within web browsers, which means that they are both platform independent and easy to integrate into other WWW-based activities.

The limitations that are placed on interactions with applications when using the World Wide Web are discussed in more detail in Section 4.3.2.

3.5 DESIGN RATIONALE SYSTEMS

Design Rationale systems are methodologies and applications which support the construction of a rationale. Rationales are semi-formal descriptions and explanations of the choices that lead to a particular design being chosen for an artefact. This section discusses some of the different methodologies that have been used. A good summary of evaluations of design rationale methodologies and systems can be found in Buckingham Shum & Hammond (1994).

Design rationales grew out of Toulmin's (1958) work on argumentation, and adopt similar graphical semi-formal notations. Toulmin developed argumentation structures for representing the reasoning behind a claim, while design rationales focus more specifically on the reasoning behind the creation of an artefact. Design rationales themselves and the process of creating them are intended to:

- Structure and analyse novel design problems
- Keep track of decisions
- Communicate design reasoning within projects
- Maintain consistency in decision-making
- Track progress in projects and identify recurring and unresolved issues
- Support the building of cumulative design knowledge, through reusing design rationale
- Assist the integration of perspectives from multiple stakeholders on decisions

(Buckingham Shum, 1996)

Shipman & McCall (1997) separate these varying goals into three overlapping perspectives to design rationale.

- **Argumentation:** design rationale records the reasoning behind design decisions and can be used to improve the decision-making process. The process of design focuses on the construction of the design rationale.
- **Communication:** design rationale records the naturally occurring communication between members of a design team for future reference. Design Rationale systems are simply CMC applications which archive communication.

- **Documentation:** design rationale records meta-information about design decisions such as when they were made, by whom and why, for those outside the project to view. To a certain extent, Design Rationale systems formalise the production of reports.

Within the argumentation perspective, which is the most formal and the one I will focus on here, there are two main philosophical approaches. The first, exemplified in IBIS (see Section 3.5.1) and PHI (see Section 3.5.2), aims to document the reasoning behind a particular design. This approach focuses on the argumentation produced when multiple designers collaborate. The second approach, exemplified in QOC (see Section 3.5.3), aims to explore and map the range of possible designs in a Design Space Analysis (DSA: MacLean et al., 1991). In this approach, it is recognised that different artefacts within a particular domain have different goals, which may affect the design chosen.

The basic formalism for design rationale is similar to conceptual maps, in that there are a number of *nodes*, of different types, linked together by *relations*. This formalism is particularly suited to presentation using hypertext, with design-rationale nodes mapping to hypertext nodes and design-rationale relations mapping to hypertext links (see Section 4.3 for more detail). The types that nodes and relations can take depend on the particular formalism used. A description of the main formalisms for design rationale, and the applications that use them, is given in this section.

3.5.1 ISSUE-BASED INFORMATION SYSTEM (IBIS)

As its name suggests, IBIS takes an issue-based view of design rationale, focusing on describing the process of argumentation. In his papers in the early 1970s (Rittel & Kunz, 1970; Rittel, 1972; Rittel & Webber, 1973), Rittel suggested that there were certain key issues in any design problem, which he termed 'wicked' issues. These are questions, problems or concerns that need to be discussed, whether or not they need to be agreed upon.

The IBIS methodology formalises argumentation using three types of nodes. For each *Issue*, one or more *Positions* can be taken. These are responses to the issue, which may be mutually exclusive, but are not necessarily so. For each *Position*, there are one or more *Arguments* that might support or object to it.

During argumentation, it is intended that the participants further the argument by adding *Issues*, *Positions* or *Arguments* to the IBIS 'tree'. Once an *Issue* is raised, *Positions* can *respond-to* the *Issue*. In turn, *Arguments* can be added which *support* or *object-to* the *Positions*. New *Issues* are added in response to *Positions* and *Argument* by *questioning* or *being-suggested-by* them: they are added in response to old *Issues* in the same ways and by *generalising*, *specialising* or *replacing* them. The relationships between *Issues*, *Positions* and *Arguments* are shown in Figure 3.5. The extension of the tree can continue *ad infinitum*: there are no stopping conditions and no specified goals of the discussion.

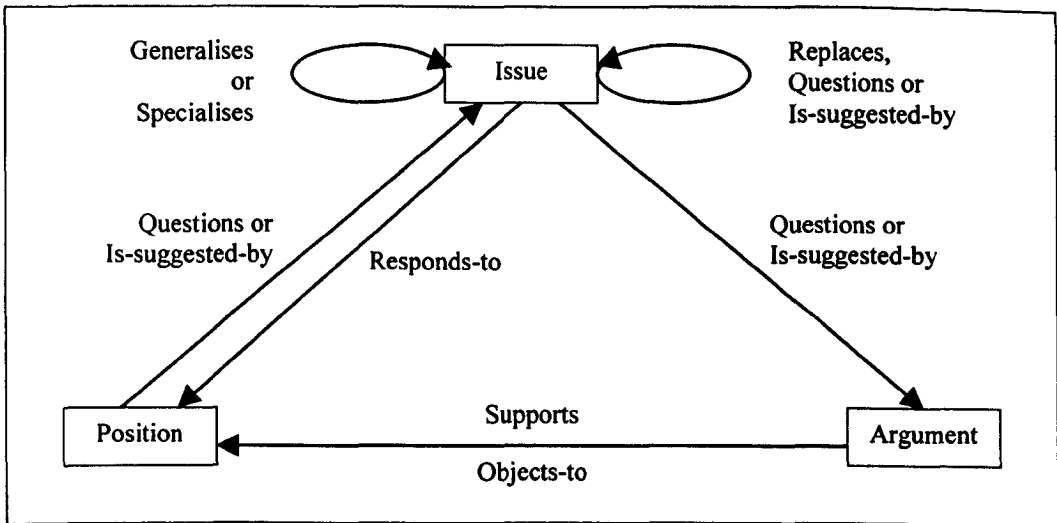


Figure 3.5: The set of legal rhetorical moves in IBIS. From Conklin & Begeman, 1988.

The IBIS representation of design rationale has been implemented in a number of systems, including itIBIS (Conklin & Yakemovic, 1991), gIBIS (Conklin & Begeman, 1988), rIBIS (Rein & Ellis, 1991), IDIS (Chung & Goodwin, 1994) and commercially in QuestMap⁹ (see Figure 3.6).

The IBIS notation can be represented either graphically, as in Figure 3.6, or textually, using itIBIS (indented text IBIS) as in Figure 3.7. Within graphical representations, a schematic view of the argumentation structure is shown (see Section 4.4.1). In itIBIS, Issues are represented by the letter I, Positions by P, supporting argument by AS, and arguments which object to the position by AO. In addition, characters were used to indicate whether an issue was resolved (*), that no decision had been made (?) or that the position had been rejected (-). While itIBIS cannot display complicated IBIS argumentation effectively, it is useful in capturing the essentials (Conklin & Yakemovic, 1991). An example itIBIS discussion is shown in Figure 3.7.

Based on the application of IBIS argumentation in both gIBIS and itIBIS, several extensions were suggested. Firstly, the IBIS representation lacks the means for representing goals or requirements. There are thus no over-arching criteria on which Positions can be judged, and no means of suggesting relevant Issues to be discussed. One suggestion is that early on in design, a brainstorming mode might be available for the express purpose of creating and refining the Issues to be discussed.

Secondly, IBIS does not offer syntactic or semantic analysis to check or support decisions: a Position could be accepted, for example, even with no supporting arguments. Ideally this type of inadvertent action should be spotted and prevented. Additionally, semantics checks could keep the network up-to-date (by removing arguments which no longer apply), and ensure that

⁹ QuestMap is available from Group Decision Support Systems, Inc. at <http://www.gdss.com/>


```

*I: Which processor should be used?
  ?P: A.
      AS: Fast.
  *P: Processor B.
      AS: Already in use, thus cheaper.
  -P: Processor C.
      AO: Will not be available in time.
          *I: Can it be delivered sooner?
              *P: No.
                  AS: Design will not be completed til next year.
  
```

Figure 3.7: An example itIBIS discussion. The root issue (I) is marked as resolved (*I) and is addressed by three positions (P), the second of which is selected (*P). The positions have various supporting (AS) and objecting (AO) arguments. The argument objecting to the third position has a subissue that seeks verification of the argument's assertion. From Conklin & Yakemovic, 1991.

In IDIS (Integrated Design Information System), Chung & Goodwin (1994) added three new types of nodes to the IBIS formalism: *facts*, *comments* and *decisions*. They also added four types of links: *copyOf* to indicate when two nodes are identical; *followUp* for the reopening of issues; *combinedWith* to combine several positions together; and *replaced* to indicate where several issues replace one issue or vice versa.

However, even when implemented in the unsophisticated form of itIBIS, the IBIS method of design rationale representation proved successful on several counts (Conklin & Yakemovic, 1991). Firstly, the record of project rationale was particularly useful during changes in the group, and in explaining to new members why a particular action was taken. Secondly, during the requirements analysis process, the IBIS methodology exposed faulty assumptions and missing reasons for requirements, and thus helped to clarify them. Thirdly, during the design process, the methodology helped in identifying a number of errors and problems within the design, thus saving the time and money that would have been spent fixing them later. Fourthly, recording communication using the IBIS representation made for more productive and focused meetings.

3.5.2 PROCEDURAL HIERARCHY OF ISSUES (PHI)

PHI is another process-oriented approach to argumentation. As with IBIS, the main backbone of the PHI representation is the expression of a number of issues, with possible positions being supported or objected to by a number of arguments. However, PHI adds two important pieces of information to the structure: how positions depend on each other: and issues which are not necessarily contentious, but nevertheless have an impact on the eventual design (Fischer et al., 1991). PHI represents these two issues by replacing the inter-issue relationships of *questioning*, *being-suggested-by*, *generalising*, *specialising* and *replacing* with, simply, *serves*. In this context, issue A *serves* issue B if the resolution of A affects the resolution of B. The most often used *serves* relation is *is-a-subissue-of*. This is used when solving one issue is a subtask of resolving another.

These alterations mean that PHI networks are less complex than their comparable IBIS networks, since links of different types do not need to be represented. In addition, the network stems from a single node, the overarching issue to be addressed, such as 'how should this kitchen be designed', and all issues follow from this in a quasi-hierarchical structure.

The application of PHI is not only its representation to the user, but its inclusion in a complete design package. PHI has been implemented in several packages: for example JANUS, aiding in the design of kitchens (Fischer et al., 1991); and NETWORK-HYDRA, aiding in the design of computer networks (Fischer et al., 1992).

The philosophy behind the applications of PHI is that communication within a group of designers should be indirect, and therefore not interrupt their primary function, which is design. These applications are therefore entire design environments, in which not only the argumentation side of design, but also construction, can be completed. This is beneficial in that the process of producing a design rationale is less expensive for the users, as it can be incorporated into their usual work.

Viewed from a theoretical perspective, the incorporation of design rationale during actual design is also attractive. Design can be viewed as situated action, and as non-reflective until a breakdown occurs. That is, designers do not reflect on *why* they are designing in a particular way until that design comes under question from other designers or fails to match requirements in some way. These applications attempt to integrate the construction (action) side of design with the argumentation (reflection) side of design by informing the user when a breakdown occurs (Buckingham Shum et al., 1997).

The general architecture for these applications involves a Construction Kit, Argumentative Hypertext, a Catalog, and Specification and Simulation Components. The Construction Kit is the medium for implementing design, including a palette of items for use in the design. The Argumentative Hypertext is a PHI network of issues, positions and arguments, which can be added to by the user as they wish. The Catalog is a collection of example designs. The Specification Component allows designers to specify some characteristics of the design, and is used to retrieve catalogue items and filter the hypertext. The Simulation Component allows for experimentation during the design process (what-if games).

Three mechanisms support the integration of these components. These are the Construction Analyzer, the Argumentation Illustrator and the Catalog Explorer. The Construction Analyzer is essentially a critiquing system, which indicates to the user when a breakdown occurs, and points the user to the relevant piece of argumentation, indicating why the design is non-compliant with the argumentation. The Argumentation Illustrator illustrates the argumentation using examples drawn from the catalogue. The Catalog Explorer helps the user to find example designs relevant to their own from the catalogue.

JANUS and NETWORK-HYDRA are essentially two implementations of this architecture in different domains. There has been little empirical testing of the architecture in these domains, but it does have some promising attributes. To begin with, the architecture for the design environment is generic, and therefore could conceivably be used in any domain. Related to this, the argumentation hypertext used during the design process could also be reused in a different context. Thus, these systems are conceivably reusable. However, there is a problem in that the argumentation that is built up during one project is not necessarily relevant in another project. This means that the argumentation hypertext may become contaminated with knowledge that is not relevant, affecting its reusability.

In JANUS, the argumentation hypertext was not extensible by designers using the system: the users could not alter the hypertext or add their own expertise to the critiquing system. In NETWORK-HYDRA, however, users could add their own rules to the critiquing system. The advantage of user extensibility is that the argumentation adapts to the project being undertaken and only needs to be seeded, not fully built, before construction can start. However, as well as the implications for reusability of this approach (indicated above), this brings with it the possibility of inter-personal arguments that may adversely affect the design and compromise the effectiveness of the critiquing system. This relates to the problems of multiple experts in the construction of ontologies described in Section 2.5.2.1.

3.5.3 DESIGN REPRESENTATION LANGUAGE (DRL)

DRL was developed as a more expressive language for representing design rationales. Lee & Lai (1991) identify a number of different meanings for design rationale, such as the historical record of a system, the set of psychological claims embodied by an artefact, or a possibility space of alternative designs for an artefact. They suggest that these meanings can be placed in a series of five progressively richer models of design rationale, described below. These models work by picking out different types of arguments from the overall design rationale.

DRL in application (e.g. SIBYL: Lee, 1990) supports the creation, editing and browsing of a number of objects. Objects of the type *Is Related To* and its subtypes act as links between other objects. Legal objects that can be linked depend on the type of the relation.

The five models and their related objects are:

- **Argument space**

The first model is a representation of *all* the reasons that are related to an artefact: that is, all the reasons that contribute to why it is constructed in the way it is. These reasons may be historical and/or logical reasons for the artefact, and the degree to which the reasons are related may be specified to a variety of levels. Arguments are represented using a set of

related Claims, each of which has an associated plausibility, which indicates how much the claim is believed. Claims can be related to each other through objects such as Supports, Denies and Presupposes. Since all relations are subtypes of Claims, each can be argued about.

- **Alternative space**

The second model makes alternatives explicit and relates arguments to them. The relationships between alternatives are also made explicit, as is their present status (e.g. rejected). The *alternative space* allows for some operations, such as being able to compare alternatives, and for argumentation about why certain alternatives are being discussed. The alternative space is constructed by linking Alternative objects using *Is A Kind Of* relations. The alternatives can be argued about, as can the relations between them, since each relation is a Claim.

- **Evaluation space**

The third model introduces an *evaluation space*, in which the status of different alternatives is made explicit. Here, argumentation about the present evaluation of alternatives occurs. The evaluation space is represented by each Claim having an associated evaluation attribute. The evaluation attribute of a claim is a combination of its plausibility (how likely the claim is true) and degree (to what extent the claim is true). The evaluation given to an alternative is measured in the *Is a Good Alternative for* relation between an alternative and a decision problem, and thus a combination of the *Achieves* relations between the alternative and the decision problem's subgoals.

- **Criteria space**

The fourth model introduces a *criteria space*, in which the criteria on which evaluations are made are themselves made specific. Here, discussions about the criteria on which alternatives should be judged can take place. The criteria space contains Goal objects, of which Decision Problems (goals of choosing the best alternative) are a subtype. Goals can be related with the *Is A Subgoal of* relation (which again can be argued against, since it is a Claim), and can be grouped together to represent the various types of relations that sub-types may have.

- **Issue space**

Finally, the fifth model introduces an *issue space*. Each issue has an argument, alternative, evaluation and criteria space related to it, and the relations between the issue spaces are made explicit. DRL attempts to represent argumentation in all these spaces. Individual Decision Problems and the relations between them make up the Issue space.

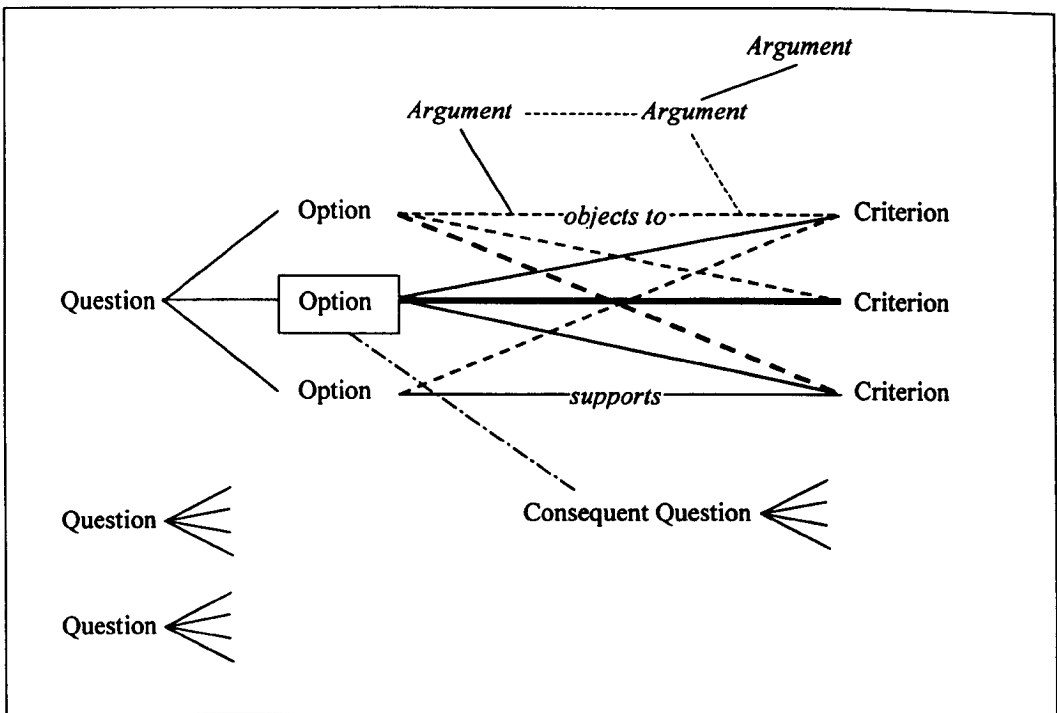


Figure 3.8: The generic QOC notation as used in Design Space Analysis. Note the use of link-thickness to indicate relative weights of Assessment. From Buckingham Shum, 1996.

3.5.4 QUESTIONS, OPTIONS AND CRITERIA (QOC)

QOC is proposed as a method by which a Design Space Analysis (DSA: MacLean et al., 1991) can be carried out. A DSA seeks to locate an artefact in a possibility space and discover why that artefact was chosen out of those possibilities. Thus, the DSA has to structure the design space in such a way that the alternatives, and the criteria for choosing between them, are explicated. The process of Design Space Analysis using QOC consists of five phases (MacLean, Bellotti & Shum, 1993):

1. Identify relevant information
2. Structure material into rough QOC
3. Flesh out design space
4. Reformulate design space to tidy it up
5. Make design decisions

The QOC representation consists of a number of Questions that are each answered by a number of Options, which are judged against a number of Criteria. The generic notation is shown in Figure 3.8. The Questions that are chosen are vital to the success of the QOC representation, and are interactively refined and reformulated during the design process (Bellotti, MacLean & Moran, 1991). Questions are refined to structure the design space better, for example by being worded so

that Options at only one level of abstraction are discussed, by making explicit any assumptions, and generally by altering the perspective on design.

Unlike in IBIS or PHI representations, Criteria are used instead of Arguments to judge between the Options available. Criteria relevant to a particular Question are linked to *all* the answering Options, either positively or negatively, which makes the Options more easily judged against one another. In addition, Criteria across the entire design space can be examined when looking at *why* the artefact was designed in a particular way, and are thus more useful than the specific Arguments used in IBIS or PHI. Arguments, in terms of IBIS or PHI representations, are simply labels for links, not bases for evaluation themselves. Each link can also have an Assessment, which gives a measure of how true or important the link is.

MacLean et al. (1991) remark on the unique characteristics of the QOC representation. Firstly, the emphasis is on a space of possibilities, rather than on developing one single design. Secondly, there is a focus on Criteria and therefore goals, which makes evaluation of the design much easier. The DSA is not, however, a record of design, but a stylised representation of the design space. Therefore, the criteria, questions and options are not necessarily *all* those discussed during the design, but a condensed history. The QOC representation is intended to be used during the design process to aid the 'reflective' part of design, rather than the production of an artefact.

The QOC representation, like IBIS, contains no stopping conditions, and is eminently expandable. Questions can be posed to any level of detail (though, for good structuring, general questions are preferred). Equally, any relation can be argued about. There is no searching for the truth of the design, but rather an expansion of the designer's views in an attempt to counteract the confirmation bias usually experienced during design (Bellotti, 1993).

Design Space Analyses have been used in a number of domains (e.g. the design of Automated Teller Machines (MacLean et al., 1991), User Interface Design (Bellotti, 1993) and Hypertext Application Design (Shum, 1993)). These suggest that by using the QOC representation, designers could improve their reasoning by structuring their thinking more logically. The QOC representation can also aid in brainstorming through Options and Criteria, and by integrating theoretical and practical perspectives on design. Finally, research on QOC has looked at how the QOC representations can be referred to *after* design (an area that is seldom examined). This research suggests that the representation can aid in the rebuilding of design rationale after the system is built.

3.6 CONCLUSIONS

This chapter has examined the issues involved in the design of computer-mediated communication applications and touched on some of the areas in which they differ from each other. I have then focused on two technologies that have influenced the design of APECKS, which is described in Chapter 5.

The first requirement for an application for the support of collaborative work is simply that it allows communication between members of the group. Indirect communication can be used without breaking the normal workflow of an individual, and is therefore most useful when the task normally requires solitary activity. Other activities, including socialisation, which is an important part of collaborative work, require synchronous communication. It is therefore one of the requirements for APECKS (requirement 15—'Users should be able to communicate synchronously'—of Section 5.3.3.4). For this type of communication, context, in the form of a virtual environment, may be beneficial, and with current technology, text is the only practical medium.

In Section 3.4.2, I identified a number of desirable features for CMC applications. Those requirements were:

- **Accessible:** users should be able to access the system with their current technology.
- **Learnable:** users should be able to learn how to use the system within a short period of time.
- **Persistent:** users should keep the same identity across many sessions.
- **Archivable:** users should be able to reread or replay previous communication.
- **Authorable:** the system should be easily initialised with information to be made available to users.
- **Programmable:** the system should enable organisers to automate appropriate tasks.
- **Reusable:** the system should be reusable in many situations.

From Section 3.4.2, we can see that MOOs meet most of the above requirements without any modification. As discussed in Section 3.4.3, adding a WWW interface to the text-based virtual environment enhances both the accessibility of the CVE and also its learnability, as users are able to access the system with their current computer system and applications. In Section 5.4.1, I discuss how WWW-enhanced MOOs were used as the basis of APECKS. The requirement for persistency of user identity is given for APECKS as requirement 1 ('Users should have persistent identities') of Section 5.3.3.1, while the requirement for the archiving of communication is given for APECKS as requirements in Section 5.3.3.4:

- Discussion between users should be archived for future use (requirement 17)
- Discussion archives should be searchable (requirement 18)

The work on design rationale is relevant to the design of APECKS in two ways. Firstly, it highlights some of the issues involved in structured, indirect and asynchronous communication. The work on structured communication shows the benefits of prompting for certain actions to be taken at certain points (as well as the disadvantages). This leads to requirement 12 ('Prompts should be generated for users to make changes to and discuss roles on the basis of comparisons') of Section 5.3.3.3. However, the above discussion of various design rationale formalisms shows that there is a large variety of types of utterance used, depending on both the aim of the design rationale and the context in which it is used. I will return to this point in Section 5.3.3.4 and Section 8.4.3.

Secondly, as we shall see in Section 5.2, the main process supported by APECKS is one of *design*, and more specifically Design Space Analysis. The applications of PHI (Section 3.5.2) show how design rationale can be incorporated into a design package. The main requirements for APECKS arising from this are discussed in Section 5.3.3.5. In addition, the work on DRL (Section 3.5.3) and QOC (Section 3.5.4), and the enhancements made to IBIS (Section 3.5.1) show that a representation of the goals and assumptions behind a design are as important as the various options and argumentation that surround them. This leads to the following requirements in Section 5.3.3.4.

- Evaluative criteria should be represented within the system (requirement 13)
- Arguments concerning the validity of criteria and the applicability of criteria to roles should be represented within the system (requirement 14)

As indicated in the discussion of IBIS (Section 3.5.1), it is helpful to have a link between the artefact as constructed and the rationale behind that design. This leads to the following requirement in Section 5.3.3.4:

- Users should be able to indicate areas under discussion directly within communication (requirement 16)

One of the advantages of computer-mediated communication is that it can be archived both automatically and in a format that is relatively easy to index and reference into (unlike video or audio). In particular, one advantage of structured communication such as design rationale is that the archive produced is organised in a logical and searchable manner. The issues concerning the retrieval of this information from an archive, or, in indirect communication, from the intermediary (in this case, a computer) are discussed in the next chapter.

CHAPTER 4 INFORMATION RETRIEVAL

4.1 SUMMARY

Support for collaborative ontology construction involves the creation and maintenance of two types of structures: ontologies and archived communication. As well as building these structures, those using such a system have to be able to navigate through it in order to find knowledge structures or annotation threads. In this chapter, I discuss the retrieval of information from these kinds of structures: hypertext systems.

Hypertext systems provide several advantages for organising information used in distributed collaborative work in organisations, namely:

- Representation of large amounts of information
- Multiple types of information
- Easy addition and alteration of material
- Use in extended tasks
- Easy navigation by non-experts
- Multiple perspectives

Information retrieval tasks can be classified in terms of the extent to which users know what they want to find out. Highly goal-directed inquiries are termed searching while those in which the user has little idea what they are looking for are termed browsing. The challenge for hypertext system designs is to support the full range of information retrieval tasks.

This area has been of particular interest over the last few years due to the emergence and popularity of the World Wide Web. The World Wide Web is an extremely large and unstructured hypertext system, which presents difficulties when attempting to retrieve information from it. In this chapter I discuss some of the solutions that have been put forward to attempt to present the WWW in more structured and navigable ways, both through the use of metaphor within spatial representations and the automated construction of schematic representations.

I finally address some of the issues surrounding collaborative information retrieval. Firstly, information from other users of the system can be used to aid in single-user searches. Secondly, users can collaborate while carrying out a search. I discuss some of the search support systems that address these issues.

4.2 INTRODUCTION

Computer-based communication tools offer an important advantage over non-computer-based communication: they are inherently archivable. However, the huge amounts of information stored lead to a further problem: how to retrieve the necessary information from within it.

There are many different types of information resource. Firstly, we can classify information resources in terms of the media which is used, be it text, graphics, audio or video. As we have seen in the discussion of computer-mediated communication in Chapter 3, some media are inherently more expressive than others, and can therefore hold different kinds of information. Hypermedia information resources contain information in many different media. The medium of an information resource determines how easy it is to discover what information it holds and to index into it. For example, video currently requires a human to watch and classify its contents, whereas text can be scanned automatically by a computer for keywords. There are efforts underway to automatically scan images for certain colours and shapes in order to identify their content (this is currently used to filter images from the Internet to prevent children accessing pornography sites).

Secondly, information resources differ in size, and this alters the efficiency of different information retrieval tactics. For example, while it might be possible to index a small hypertext system of only around 100 pages, it is not possible to do this with the World Wide Web, with millions of pages.

Thirdly, some information resources are developed top-down, with a structure imposed on them from the start, whereas others develop in a more organic fashion, without a single structure imposed on it. Both these types of information resource raise issues for information retrieval. If an information resource can be built around a structure, what kind of structure should be used in order to support easier browsing and information retrieval? If an information resource has arisen without a structure imposed on it, are there any structures that can be discovered within it instead? These issues concerning the structure of information are important not only at all grainsizes, and may be different at each level. For example, a highly structured database might be a single resource inside an unstructured hypermedia system.

4.3 HYPERTEXT SYSTEMS

The basis of hypertext is a number of nodes containing information connected together with a number of links. From this foundation, there are many differences between hypertext systems:

- **Views:** Hypertext may be browsed by viewing one node at a time, and selecting links to journey from node to node, as in the World Wide Web. Alternatively, users may be able to

see an overview of the hypertext system, showing how nodes are linked together (see Section 4.4).

- **Types:** Nodes and links may be typed in different ways. For example, in a hypertext representation of the QOC design rationale (see Section 3.5.4), nodes may be typed as 'Questions', 'Options' or 'Criteria'. Links may also indicate the relationship between the nodes being linked, such as *specialising* or *generalising*. Typing nodes and links help authors structure hypertext and give readers an idea of the purpose of the information they are viewing or about to view (Bieber et al., 1997). Some hypertext systems allow the definition of 'schemas' that define the types that nodes and links can take (e.g. Aquanet: Marshall et al., 1991; Marshall & Rogers, 1992; see also Section 8.5.2).
- **Link properties:** Links may connect only two nodes or may be able to connect more nodes together. They may be uni-directional, in which case they can only be followed in a single direction, or multi-directional, in which case all ends of a link can be used as a starting point to reach the others.
- **Link appearance:** Links may be presented embedded within the information at the linking node, or be presented separately from the node itself. For example, a link may lead from within a piece of text, graphic or video, or it may appear as a button beside the information, or both.
- **Resource combinations:** Linked resources may be included within the linking page, or may be displayed within a separate context. For example, the banner section of pages within a web site may be defined separately, but then included within each page automatically. The inclusion of sections of resources is termed *transclusion*, while the inclusion of the entirety of other resources create *composites* (Bieber et al., 1997).

Hypertext systems are popular ways of navigating information, as evinced by the number of hypertextual CD-ROMs, help systems and, of course, the phenomenal success of the World Wide Web. There are several characteristics of hypertext systems that make their use appropriate for collaborative work:

- **Representation of a large amount of information**

Organisational information systems have to contain an enormous amount of information for presentation to the user, and hypertext is well suited to store this information.

- **Multiple types of information**

Organisational information systems typically contain many types of information. Hypermedia can represent not only text but graphics, audio, and video. Hypertextual representations are therefore both rich and flexible.

- **Easy addition and alteration of material**

Since organisational information systems are constantly evolving, material will continually be added to them and changed within them. Hypertext representations allow for easy addition of material: simply by building a node and linking it to another, information can be added. In addition, hypermedia *structures* can be easily changed (Buckingham Shum, 1997)

- **Use in extended tasks**

Organisational information systems do not typically need to be read through in a sequential order. Rather, they are referred to as the user participates in extended tasks (Fischer et al., 1992). This is where hypertextual representations come into their own, as each node can be used as a starting point for accessing the rest of the system.

- **Easy navigation by non-experts**

While navigation is still a large issue in hypertext systems, hypertext does allow easy navigation by non-experts. Users of organisational information systems need to be able to navigate through information without extensive training.

- **Multiple perspectives**

Organisational information systems are typically used by a number of different users, each of whom has their own attitudes and interests. Hypertext can be a good representation in this context, because it allows multiple perspectives to be taken on the information and multiple paths to be taken through the representation.

4.3.1 PROBLEMS WITH HYPERTEXT

Hypertext, and more specifically hypermedia, is the fashionable way to present information via computer. However, research into the use of hypertext and hypermedia has revealed problems for both authors and users of hypertext systems. The most quoted problems concerning navigation in hypertext concern cognitive overhead and disorientation (or the feeling of being 'lost in hyperspace') (e.g. Conklin, 1987).

4.3.1.1 Cognitive Overhead

High cognitive overhead is a problem for both the authors and users of hypertext systems. For authors, especially of large hypertext systems, the major problem is giving a unique name to each separate node within the hypertext system. With collaborative documents, one can imagine this load increasing as authors have not only to consider their own naming of the nodes, but also other authors'. For users of hypertext systems, cognitive overload occurs due to having to keep a mental track of where in the hypertext system they are (failure to do so leads to disorientation, as described below) and having to choose between many possible links at each node.

These problems have been addressed in several ways:

- Navigation links allowing users to jump to 'home pages'
- Automatic bookmarking of pages
- History lists allowing users to retrace their steps through the hypertext
- Visual cues indicating to users where they have already been
- Guided tours of hypertext (e.g. Footsteps: Nicol, Smeaton & Slater, 1995), which allow users to gain an overview of the material before exploring on their own.
- 'Look-forward' systems that give an indication of which links are likely to be found interesting; and by providing users with cues about missed information. This reduces counter-productive exhaustive search strategies (Bell, 1995)

4.3.1.2 Disorientation

Disorientation occurs when users of the system do not know where the node they are presently viewing is in relation to other nodes. Tools that mirror those we use to prevent us becoming lost in 'real' space have addressed the problems of being lost in hypertext.

Firstly, the organisation of hypertext can be made easier to navigate. Guides on writing hypertext now focus on a not-too-flat, not-too-deep hierarchical structure (Lynch, 1995), which enable simple strategies to be used to search the entire hypertext space (similar to the 'always turn left' strategy used to navigate mazes) and to reach 'home' ('always go up'). It might also be that structures that we understand intuitively, since they reflect the real world, might be easier to navigate.

Secondly, tools can be used to give an overview of hypertext, allowing the user to discover their position in relation to other areas. Schematic overviews allow users to focus quickly in on sections in which they are interested, and give them an idea of the general structure of the document space. Systems that create or provide schematic maps of hypertext is discussed in Section 4.4.1 below.

Thirdly, authors of hypertext structures can include navigational signs to allow users to locate themselves (as well as to indicate where to go next, see below). The dual purpose of navigational signs is often neglected: to aid the user in locating themselves in the hypertext space, they need to not only know where they can go in terms of standard directions, but where these standard directions lead to. This is termed signage.

Recent thinking has questioned the notion of disorientation in hypertext being a 'bad thing'. Being lost enables us to discover new places, which we might not have otherwise stumbled across (this is termed serendipitous discovery (Twidale et al., 1996)). It may also help us to find better routes between known nodes. However, the attitude users take to being lost in hyperspace depends on

where the task they are involved in is situated on the searching–browsing continuum described in Section 4.4. Browsing can be thought of as deliberately losing yourself in hypertext, whereas becoming lost when you are desperately looking for a piece of information is very frustrating.

4.3.2 THE WORLD WIDE WEB

Hypertext has become most widely known due to its use within the World Wide Web (Berners-Lee et al., 1994). There are currently estimated to be around 13 million WWW sites, with 800 million pages being available by the year 2000. In this section, I outline the functionality of the WWW in terms of the characteristics of hypertext systems discussed above.

The WWW is a large, unstructured hypermedia system. The experience users have of the WWW is partly determined by the browser that they use and partly by the limitations of the HyperText Markup Language (HTML)¹⁰ and HyperText Transaction Protocol (HTTP)¹¹. Naturally, the standards which govern the WWW and the browsers used to navigate it are changing constantly. The discussion here details the state of affairs around the beginning of 1998: developments since then and their implications are discussed in detail in Section 8.5.

4.3.2.1 Standards

The WWW uses a client-server based protocol (HTTP) in which the client (a user's browser) makes a request to a server and receives a response. Usually, the request is for a particular file, and the response includes that file. Each request must stand completely on its own, which means that all the information needed to respond to it must be included within the request. It also means that the client must be the initiator of the interaction, which limits the role that WWW-based applications can play (Bentley et al., 1997).

Applications in which the response is not an existing file, but is generated on the fly based on the request, are becoming more common. For these kinds of applications, the request-response protocol is very limiting, as any state information must be transmitted within the request. This has led to the development of 'cookies'. Cookies are messages that encode state information within the requests and responses passed between server and client. They do not appear within the page or within the URL of the resource, and are thus invisible to the user, but are recorded on the local storage medium of the client computer and are thus accessible from session to session.

Another way around the problem of one-off requests is the use of extended connections between the client and server computers, during which a number of requests can be made and a number of responses received.

¹⁰ More information about HTML is available at <http://www.w3.org/MarkUp/>

¹¹ More information about HTTP is available at <http://www.w3.org/Protocols/>

The basis of the WWW is text, either stored within files or dynamically generated, that is marked up with HTML. HTML fulfils six purposes:

- **Indicating the purpose of sections of text**

The meanings that sections of text can have are determined by HTML. Originally, these meanings were based on the requirements of the scientific community, as these were the first intended users. Thus, as well as elements indicating document structures (like headings and paragraphs), there are others which indicate computer code, keyboard input and sample output, the kind of text common in computer science. The next step in the evolution of HTML, XML (see Section 8.5.1) allows authors to assign their own meanings to text.

- **Determining the appearance of pages**

As the WWW gained in popularity, the new users had new requirements. As the WWW became more than simply a means of sharing academic papers, the new versions of HTML gave users the opportunity to change the way their pages appeared. These included changing the colour of text and backgrounds, the size of sections of text; eye-catching effects such as scrolling or blinking text; and control of the layout of the page using of tables. This use of HTML is gradually being phased out as stylesheets (see Section 8.5.3) come into use.

- **Creating links between resources**

The real innovation of the WWW over other Internet protocols is the linking of resources together. The main kind of link used on the WWW is inline, linking a phrase within a page to another page. These inline links can also lead to other resources on the Internet or to points ('anchors') within a page.

HTML also supports out-of-line links that link the entire page to another resource. However, these are seldom used since the major browsers (Netscape and Microsoft Internet Explorer) do not display them. Instead, authors create inline links that fulfil the same purpose.

For both types of links, HTML also supports the typing of links to indicate the relationships between resources. However, again, the major browsers do not alter the display of links according to the typing, so they are seldom used.

HTML supports inline linking from areas within graphics to different resources. However, it does not support inline linking from media other than text and graphics, such as audio or video. HTML also does not support multi-directional links.

- **Inserting other media into pages**

HTML supported the insertion of graphics into pages from early on. More recently, extensions made by the major browsers gave the facility for adding other media to pages, such

as background music and inline video. In the latest version of HTML, it is possible to insert a resource in various media, and allow the browser to choose the medium in which to view it. For example, an author might supply a three-dimensional model, a video, a picture and a textual description of the moon, and the browser would display the most advanced medium it could.

- **Defining additional interaction with forms**

As the uses of the WWW increase, so has the requirement for a facility allowing users to interact with WWW-based systems. This led to the development of HTML forms, which allow the user to supply certain types of information to a WWW server. HTML forms give fairly limited interaction, although the use of scripting and Java™ applets has increased the interactivity of WWW pages (Ibrahim, 1997).

- **Giving meta-information about the page**

The final use of HTML is to provide meta-information about the content of the page. This most common use of meta-information is to provide keywords and a textual description of the page that can be used by search engines to classify the page and give users a brief description of its contents.

4.3.2.2 Browsers

The evolution of the WWW is tied closely to the evolution of the browsers used to access it. The WWW started to become popular when the first graphical browser, Mosaic, became available. At each step since then, the support that browsers offer has both extended and limited the features and usability of the WWW.

Most browsers display only one WWW page at a time and allow users to move between them by selecting one of the inline links within the page. Although some systems have been constructed which allow the automated generation of overviews, the scale and lack of imposed structure of the WWW on a wide scale mean that maps are useful only in limited situations.

Most browsers supply only limited information about the types of links, as noted above, and very few display any information about any out-of-line links that may have been included within the page. Most browsers, however, do distinguish between links that have been visited previously, and those that have not, by keeping an extensive record of the pages the user has previously viewed.

The lists of previously visited pages are kept in 'History Lists'. The length and content of history lists differs from browser to browser. Studies by Tauscher & Greenberg (1997) have shown that the stack-like mechanism used by most popular browsers for history lists is inferior, in terms of giving pages to which the user wishes to return, to simply displaying the last 6-10 URLs visited,

with duplicates removed. In addition, Cockburn & Jones (1996) showed that users did not understand the way that the stack mechanism used and therefore were frustrated at being unable to return to pages they had previously visited.

As well as history lists, most browsers give 'Back' and 'Forward' buttons which enable users to retrace their steps. Tauscher & Greenberg (1997) found that using the 'Back' button comprises 30% of navigation events within a WWW session, and the 'Forward' button almost 1%. They note, however, that a more effective history list than those currently used would decrease the efficiency of using the 'Back' button.

Finally, most browsers also support the creation of 'Bookmarks' which allow users to mark those pages to which they are likely to return and to navigate to them easily. Current browsers also often have pre-programmed bookmark lists of useful sites, especially search engines.

4.4 NAVIGATION AND MAPPING

The navigation of information resources, as with the navigation through any space, involves the user building a cognitive map of the information. It has been found that users navigating through hypertext attempt to create a survey-type cognitive map of the contents (Edwards & Hardman, 1989). Hypertext systems support navigation by supporting the creation of a cognitive map through highlighting those features that are important in navigation. For example, they may stress 'landmarks' within the system, indicate routes that can be taken through it and provide maps that summarise the content.

Smith & Wilson (1993) distinguish between schematic and spatial representations of an information space. A schematic representation presents an overview or meta-view of the information available, allowing the user to focus in on particular areas, but not supplying the actual information held. These are usually two- or sometimes three-dimensional maps of the information space. A spatial representation situates the user within the information space itself, and allows them to navigate from one area to another.

4.4.1 SCHEMATIC REPRESENTATIONS

There are many efforts ongoing to dynamically 'map hypertext' through self-organising maps both in two-dimensions and in three (e.g. Girardin, 1996; Wood et al., 1995, Benford et al., 1997). These maps are generated by calculating the 'attraction' between each pair of pages and situating those with a greater attraction to each other closer together. The attraction between pages is usually computed on the basis of the number of links between them, but could also be calculated on the degree of similarity between the text of the two pages, the number of times links are traversed between them or other measures.



Figure 4.1: Screenshot of schematic representations in WWW3D. From Benford et al. (1997).

These automated mapping systems tend to be applicable only to small information spaces. It is computationally expensive to create graphical schematic visualisations of large hypertext systems, such as the whole World Wide Web (Smith & Wilson, 1993). The resulting maps are also extremely complex and therefore cognitively demanding for the user to understand and use. The problem in these situations is reducing the amount of information to a manageable amount, for example through using fish-eye views (Feiner, 1988), which show high detail within the centre and low detail on the periphery of the view. In WWW3D (Benford et al., 1997), the amount of information is reduced by displaying only those pages that the user has visited: the representation is dynamically updated as the user moves to new pages. In WEBNET (Cockburn & Jones, 1996), the views can be filtered according to frequency of visits to pages, recency of visits and the distance from the currently displayed page: the user can decide which filter to use.

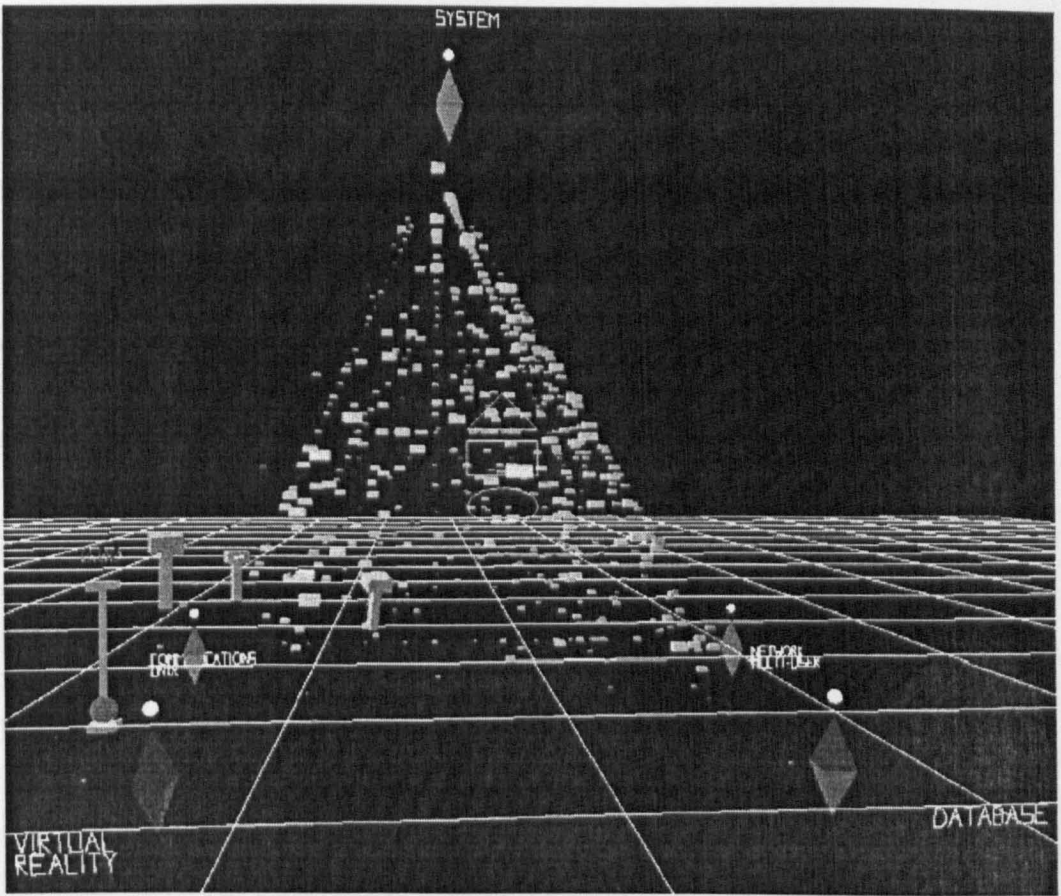


Figure 4.2: Screenshot from VR-VIBE. From <http://www.crg.cs.nottingham.ac.uk/research/technologies/visualisation/vrvibe/>

WWW3D (Benford et al., 1997: see Figure 4.1) integrates the schematic view of the WWW with a page-by-page view. Users can view the structure both from outside, as a 'ball and stick' 3D model, and from within, in which each page is represented as a sphere with links to the text of the page and images and links from the page appearing on the inside of the sphere. The schematic view also varies the colours of the balls and sticks according to the recency of the last visit or traversal, thus providing history information about the session.

In VIBE (Olsen et al., 1993) and VR-VIBE (Benford et al., 1995), the results of a set of search queries are visualised abstractly within a two or three dimensional space (see Figure 4.2). Fixed points within the space represent the queries themselves, and shapes representing the documents are arranged between these points according to their relevance to the multiple queries. The 'size' of the document representation is used to indicate the overall relevance of the document to the query set, so a number of documents do not show up at all. These visualisations allow users to gain an overview of the information available within the document space.

4.4.2 SPATIAL REPRESENTATIONS

With spatial representations of information spaces, the user is immersed within the information itself, in a similar way to the node-by-node navigation of hypertext. Spatial representations, though, attempt to enforce a spatial metaphor on the information space to aid in its navigation.

An example of this approach are virtual malls¹² within the WWW. Within virtual malls, information on various companies which are attempting to sell their products over the WWW are arranged as if they were in a mall. One page, for example, would show the 'shop fronts' while other pages would allow users to look at the range of products a shop has, or to look at one particular product. The user might have a virtual trolley in which goods were placed when selected, and so on.

Similar approaches to organising information around a spatial metaphor have been used with metaphors like museums (e.g. the British Museum¹³), cities (Dieberger & Tromp, 1993), university departments (Smith & Wilson, 1993) and so on. The information held within the metaphorical space does not necessarily fit in with the metaphor (information organised with a city metaphor does not necessarily hold information about a city, for example).

Smith & Wilson (1993) suggest that immersive spatial navigation of information is likely to be beneficial in concrete and complex domains where schematic representations are too complex or inadequate. It is hoped that within spatial representations, users can construct cognitive maps of the information in the 'natural' way that cognitive maps of real world areas are created.

An issue for these systems is how strictly to stick to the spatial metaphor being used. Within the real world, for example, searching for an item of clothing involves visiting many shops. Within a Virtual Mall, it might be possible to enter the item being looked for into a search form and be presented with summaries of the products available within all the shops. Similarly, in the real world, navigation between two places involves travel whereas in the virtual world, it is possible to 'teleport' between them.

4.5 BROWSING AND SEARCHING

Interactions with information resources can be classified in terms of the starting point of the interaction and the goal being satisfied by the interaction. Brown (1988) defined the act of 'browsing' as knowing where you are, but not knowing what you want to find, whereas 'searching' is knowing what you want to find, but not knowing where it is. These different tasks can be supported by different hypertext structures and navigational aids: searching is assisted by

¹² See, for example, The Sunnyville Mall at <http://www.sunnyville.com/>

¹³ The British Museum's home page is <http://www.british-museum.ac.uk/>

hierarchical structures and indexes, while browsing is best supported by network structures and schematic representations (Smith, Newman & Parks, 1997).

However, searching and browsing can be seen as two ends of scale rather than a simple dichotomy. It is possible to know only in general terms what you want to find: for example, 'about Granite' as opposed to 'the quartz content of Granite'. Searching and browsing are therefore simply two extremes of interactions. It should also be recognised that searching is often reliant on a degree of browsing: it is necessary to have a reasonable understanding of a domain in order to construct a search request for information in it.

4.5.1 INFORMATION RETRIEVAL ON THE WORLD WIDE WEB

The World Wide Web, as a huge, unstructured set of hypermedia, poses a real challenge for information retrieval. The popularity of the WWW means that the number of users trying to find information within it is also large, and the starting point for many interactions is some kind of search engine. There are two main ways in which information retrieval has been supported on the World Wide Web: classified directories and automated search engines (Jenkins et al., 1998).

4.5.1.1 Classified Directories

The first method for retrieving information from the WWW is through navigating through classified directories. These are hierarchical structures of topics that have been constructed by humans and populated with links and summaries of information in classified directories. The 'Virtual Library' is a distributed set of hierarchically arranged pages, each maintained by a different person who is responsible for keeping the page up to date. The first use of such a hierarchy on a single site, though, was with Yahoo¹⁴. On a smaller scale, individual sites often create hierarchical structures for their pages, and the major web browsers allow users to construct their own hierarchies in the form of bookmark lists.

Users find information in classified directories by, at each level, selecting the most appropriate subsection until they find a list of pages that apply to the topic area. While these structures are easy for novices to navigate and are less likely to contain irrelevant pages, Lindop et al. (1997) report four disadvantages due to their construction by humans:

- **Poor coverage:** the directories can only contain information about pages that the constructor of the directory knows about and has classified. Where the directory attempts to cover the entire WWW, the task of reviewing and classifying all the pages is impossible, both in terms of the identification of new pages and the number of pages on a given topic. In Yahoo, new pages are identified by having authors of pages submit their pages to be listed within a

¹⁴ <http://www.yahoo.com/>

particular topic, and reviewers examine them for suitability, thus restricting the scope to only those pages which meet reviewer's criteria.

- **Out of date information:** the content of the WWW changes daily, with pages being removed, moved and new pages being added. This leads to maintenance problems as each link has to be regularly checked to see if it still leads to the same resource. This is a particular problem for classified directories as a human usually carries out the maintenance. Maintainers often rely on those using the directory to inform them when a link is out of date.
- **Constrained query results:** the types of queries that can be made of classified directories depend on the classification structure used, both in terms of content and of grainsize. These constraints can be good in that they enable new users to gain an understanding of the scope of the directory, but they cannot be used as flexibly as keyword searches.
- **Biased information:** the maintainer of a classified directory determines which pages are listed within the directory and which are not, and how the directory is structured as a whole. Bias may occur quite innocently in how the pages are selected, for example by only including pages written in English simply because those are the only ones the maintainer understands. Also, the hierarchies that are used within classified directories may be generated on the basis of either the popularity of areas to the prospective users of the directory or the maintainer's view of the taxonomy of the domain (i.e. its ontology, see Section 2.3). Either of these structures may be inappropriate to some users of the hierarchy.

4.5.1.2 Automated Search Engines

The majority of search tools use automated 'spiders' or 'robots' to navigate through the WWW and locate pages. These pages are then summarised in terms of keywords used within them, and metadata about the page, such as the file type, where it comes from and its description, if one has been given, is stored within a database. The robot then moves on to other pages linked to by the page analysed.

Users construct logical queries and retrieve information from the database listed in terms of how well it fits the query. Automated indexing of the WWW in this way gives much more up-to-date information about the contents of the WWW (although it can never be complete as the WWW is constantly changing). However, queries often give a huge number of irrelevant 'hits', the queries are hard for novices to construct, and the information accessible through them is sometimes of poor quality (Jenkins et al., 1998).

The methods by which documents are ranked according to a search vary from search engine to search engine. In a study of the most popular WWW search engines, Pringle et al. (1998) found that most identify keywords within the title, headings and metadata. Recently, search engines have started to discriminate against pages with a high amount of repetition of keywords due to an

increasing tendency for authors to artificially repeat keywords (known as 'spamdexing': Marchiori, 1997).

Marchiori (1997) suggests the use of 'hyperinformation' in identifying relevant pages from a search query. Although some search engines take account of the visibility of a page (the number of pages that link to the page), visibility is not an indication of the quality of a page, merely its popularity. Hyperinformation takes into account the content of other pages linking to and linked to from the page in order to assess its quality.

As the World Wide Web develops, more tools are made available for search engines to use in identifying the true content of pages. Some of the newer developments and their implications are discussed in Section 8.5.

4.6 COLLABORATIVE INFORMATION RETRIEVAL

The vast majority of work on information retrieval concentrates on users working as individuals to find information. Indeed, the vast majority of systems that are used to access information are designed, if not for one user at a time, then for users to be unaware of each other. However, this ignores the way that we actually use information, especially when we work collaboratively, but also when we work alone. We can divide the need for multi-user information retrieval systems into two sections: its use in single-user searches and its use in multi-user searches.

4.6.1 SINGLE-USER SEARCHES

The most efficient intelligent agent for filtering useful from irrelevant information is often a person like us in every way (with the same interests, perspectives, preferences and so on) who has already looked at and processed the available information stores. We are more inclined to ask the person we share a room with whether they know any good papers on a particular topic than we are to wrestle with the (single-user) CD-ROM.

A common assertion about the Internet is that 'there is a place somewhere out there with information about X'. However, it would be more appropriate to say that 'there is someone out there who knows about X': the vast body of postings to Usenet News groups is testament to this. The challenge to information systems is to allow those who search for information on their own to access the searches of other, similar people. Supporting direct or indirect communication between users can do this. Individual users might also benefit from meta-level information about search strategies used by other (more experienced) users, as well as from the results of such searches (Twidale, Nichols & Paice, 1996).

One disadvantage of systems that enhance a user's search by using others is that those helping the user have no benefit from doing so. Automating the collection of search information from them can reduce this problem. For example, Wittenburg et al. (1995) use the collated web browser

bookmark lists of a collaborating group to create combined subject trees. These trees structure WWW links in ways that others have found useful, so that the user can take advantage of the structuring that others have carried out.

4.6.2 MULTI-USER SEARCHES

When we work collaboratively, the need for support for multi-user information retrieval is doubly clear. Information resources are often used collaboratively even when they are not designed to be used in that way: books are shared; library information system screens are gathered around; and groups of tourists wander around cities. In these cases, the product at the end of the search is mutually beneficial to all those who take part in the search.

There has been relatively little research into multi-user search strategies. One might presume that the most efficient strategy is one of 'divide and conquer', whereby the group arranges first who is to search where and for what, they then split up, and later regroup to pull together the information they have each individually found. However, it is likely that social factors would disturb this strategy, and that, unless there was good support for communication which is entered into by all participants, there would be duplication of effort.

Within Cobrow (Wolf & Froitzheim, 1998), other users navigating the same area of the WWW are visible and can be communicated with, using text or audio. This, Wolf & Froitzheim suggest, decreases the 'loneliness' of the WWW. They translate WWW pages into locations within the virtual space, and the links between them as paths that can be navigated. The visibility of other users depends on how far away they are within this virtual space. A similar approach is taken with WWW3D and the Internet Foyer (Benford et al., 1997).

Within VR-VIBE (Benford et al., 1995), multiple users can interact with a three dimensional visualisation of a set of queries. VR-VIBE presents the document space within a virtual environment, in which users are represented as avatars. Users can interact within VR-VIBE both directly and indirectly. Direct communication occurs either synchronously, through an audio channel, or asynchronously, through annotations which can be attached to the representations of documents and search queries within the virtual space. Indirect communication and awareness of other users occurs through the coexistence of the users within the environment. Any user may change or move the queries, or may mark documents as interesting or boring. It is hoped that through these interactions, users can work together in navigating through a document space.

Masinter & Ostrom (1993) developed a mechanism within a Collaborative Virtual Environment (actually a MOO, see Section 3.4) to support collaborative information retrieval with Gopher. In their system, Gopher menus were displayed within the virtual environment, and anyone within the same virtual room could interact with them. This enabled people to collaborate in constructing searches and facilitated the sharing of and talking about information found in this way. Similarly,

Juggler (Dieberger, 1997) linked a MOO to the WWW by allowing people, rooms and objects within the virtual environment to 'pop up' pages within a user's web browser, thus helping users chat about material on the WWW.

Research by Austin, Liker & McLeod (1993) on the control of group support systems suggests that those members of groups with more influence or status and those with higher technological competence will tend to dominate use of such systems. They also found that those groups that appointed a single controller of the support system performed slightly better, though were less satisfied than those groups in which anyone could use the system. However, these findings were for a system designed for sharing information during group meetings, and may not be applicable to information retrieval tasks: there has been little evaluation of the kinds of applications described above.

4.7 CONCLUSIONS

This chapter has given a brief overview of the issues surrounding information retrieval. I have focused principally on hypertext systems, and the World Wide Web in particular, as these systems provide advantages for distributed collaborative work in organisations, namely:

- Representation of a large amount of information
- Multiple types of information
- Easy addition and alteration of material
- Use in extended tasks
- Easy navigation by non-experts
- Multiple perspectives

Two views can be taken of a hypertext system. The first view involves a schematic representation of the information within it, in a two- or three-dimensional map. The second view is of the information itself, and involves navigating from node to node. Applications that support schematic representations can generate maps automatically, focusing in on particular areas of the hypertext. Views of the information itself often involve spatial representations, or the use of spatial metaphors for representing the information. Spatial representations of information may aid the navigation of complex concrete domains.

The task of gathering information from within a hypertext system can be situated on a spectrum ranging from searching, in which the user has a very specific goal in mind, to browsing, in which the user may have no particular goal in mind. Hypertext systems should support all types of information gathering to maximise their value to users.

Information retrieval from the World Wide Web is difficult due to its size, the unstructured nature of the information within it and the fact that it is constantly changing. Browsing through the WWW is very easy due to the simple hypertext interface. Searching involves navigating to particular sites that provide overview information of the WWW. There are two types of such sites:

- **Classified directories**, which give hierarchical classifications of pages
- **Automated search engines**, which allow users to query a database of WWW pages

In collaborative work, information retrieval can be supported both on a single-user basis and on a multi-user basis. The search results of others within a collaborating group can benefit each member of the group both by supplying the information that the member needs and by suggesting search strategies. Several systems have been constructed which support the synchronous co-operative navigation of online information.

Two important issues arise from this chapter:

- **The need for structured information**

Hypertext information can be structured in terms of the organisation of links between nodes and the structural mark-up of the information within the nodes.

Giving structure to the organisation of a hypertext system means that schematic representations of its content are less complex and therefore easier to understand. Organisational structures also make the hypertext more predictable and therefore easier to navigate. Organising information around a metaphor may be one way of giving it structure.

Using structural mark-up within nodes enables the automatic identification of the meaning of a piece of text, which enhances the searchability of hypertext. For example, searching for someone with the surname "Fish" on the WWW would result in a barrage of information about fish. If the purpose of a piece of text is indicated, it would instead be possible to search specifically for people with the surname "Fish" rather than relying on keyword searching.

- **The need for collaborative support in hypertext**

There is little widespread support currently for collaborative information retrieval from hypertext. Giving access to information structured by others enables the user to capitalise on the work already carried out by others. Synchronous collaboration in information retrieval may enhance the search process.

Chapter 3 and this chapter have outlined some of the issues concerning distributed collaborative work as a whole. As a type of distributed collaborative work, ontological engineering has the

same problems and can use the same solutions. In the next chapter, I describe a system that has been designed to support collaborative ontological engineering by using these techniques.

CHAPTER 5 APECKS SYSTEM REQUIREMENTS AND DESCRIPTION

5.1 SUMMARY

In Chapter 2, I outlined some of the issues in knowledge engineering and specifically focused on the problem of the collaborative construction of ontologies, arguing that ontologies are designed rather than discovered. Chapter 3 looked at the central aspect of collaborative work, communication, and investigated some methods for supporting the design process. Chapter 4 discussed the retrieval of information from the structures built during the collaborative ontology construction process. In this chapter, these three areas are brought together into a system named APECKS (Adaptive Presentation Environment for Collaborative Knowledge Structuring).

Taking a constructivist viewpoint, APECKS aims to support the Design Space Analysis of ontologies within a particular domain. It does this by supporting a methodology involving a cyclical process of:

- **Construction** of personal knowledge representations by each user, with support for direct knowledge acquisition from domain experts as well as standard ontology editing facilities for knowledge engineers.
- **Comparison** of these personal knowledge representations to identify similarities and differences between them.
- **Discussion** of the knowledge representations and their comparisons, including the identification of evaluative criteria against which the representations can be judged.

Throughout the process, any changes that are made to the knowledge representations are recorded. These can be annotated to create a rationale for the design of a particular representation.

This chapter discusses the requirements of each activity within APECKS and describes how these requirements were met in the working system. Chapter 6 outlines several scenarios in which APECKS could be used while Chapter 7 discusses its evaluation.

5.2 INTRODUCTION

Chapter 2 introduced the problem of constructing ontologies for the sharing of knowledge between knowledge engineers and, within corporate memory systems, those who are not knowledge engineers. In Chapter 2, I argued that ontology construction and maintenance should be seen as a collaborative design process, due to the variety of conceptualisations of domains held by experts and the variety of methodologies used by knowledge engineers. Chapter 3 and Chapter 4 elaborated on some of the requirements of a system that would support such a process.

This chapter describes a system constructed during the course of this research: APECKS (Adaptive Presentation Environment for Collaborative Knowledge Structuring). The files necessary for running APECKS are given on the attached CD-ROM; the README from the CD-ROM is given in Appendix I. APECKS was built as a proof-of-concept: to investigate the feasibility and utility of taking a constructivist perspective in the support of the distributed collaborative construction of ontologies by knowledge engineers, and directly by domain experts. To break down what this acronym contains:

- **Knowledge**

Knowledge is the central component of Computer Supported Collaborative Work (CSCW), hypertext and (obviously) ontology engineering systems. Within ontology engineering (see Chapter 2), the emphasis is on acquiring knowledge from experts and then structuring it into a reusable and sharable representation. Within CSCW systems (see Chapter 3), the emphasis is on sharing knowledge between members of the collaborating group. Within hypertext (see Chapter 4), the emphasis is on retrieving knowledge from the system.

- **Structuring**

Structuring involves arranging information into a meaningful organisation. Structure has come up in all three of the preceding chapters:

- **Structured knowledge:** Knowledge representations give structure to information, with formal knowledge representations imposing more structure than informal ones (see Section 2.7).
- **Structured communication:** Structured communication in Computer Mediated Communication (CMC) systems mean that they can prompt for certain types of message (see Section 3.3.2). It also means that the communication is more easily archived and searchable by computers.
- **Structured design space:** The task of Design Space Analysis (see Section 3.5.4) is to structure the design space so that the advantages and disadvantages of a particular design can be easily ascertained.

- **Structured hypertext:** Support for information retrieval within hypertext through both spatial and schematic representations requires a structure to be used or discovered (see Section 4.4). Structures within hypertext help users navigate as they can predict where links go.

- **Collaboration**

At the most basic level, collaboration involves communication between members of a group, and this is what CMC systems focus on (see Section 3.3). Collaboration within hypertext systems is gradually being recognised as important in supplying users with information about others' searches and search strategies, reducing the 'loneliness' of navigating the World Wide Web (WWW), and in the information retrieval process itself (see Section 4.6).

Within knowledge engineering, collaboration has focused on the supply of reusable knowledge to the community as a whole (see Section 2.6). From the classical ontological engineering perspective, this is the only context in which collaboration takes place, since there can be only one accurate knowledge representation of a domain. From a constructivist perspective, however, there is recognised the need for collaboration in the construction of the ontology itself (see Section 2.4 for a discussion of classical and constructivist approaches).

Collaboration involves more than just communication, though. Collaboration involves coming to a shared understanding of the problem area, including what the goal of the collaboration actually is. Part of reaching that shared understanding involves the exploration of alternative viewpoints, which is what Design Rationale systems (see Section 3.5) do so well.

- **Environment**

Any CSCW tool provides an 'environment' in which the users operate. What I mean here is a virtual environment with which embodied users can interact. Collaborative Virtual Environments (CVEs: see Section 3.3.1) have two benefits for organisational memory systems:

- CVEs provide a context within which people can communicate. This context can aid collaboration both by providing status cues and by allowing users to bring in additional information, such as by displaying a diagram on a virtual whiteboard.
- CVEs provide a spatial organisation of information that can supply cues to users, allowing them to navigate more easily (see also Section 4.4.2).

- **Adaptive Presentation**

Different users prefer different presentations of information because they have different interests, abilities (Fink, Kobsa & Nill, 1997), different preferences and conceptualise the

information in different ways. It is only through understanding a user's abilities, preferences and conceptualisation of the information that it can be presented to them in the way they require. 'Virtual documents' suited to the user can be dynamically generated on the basis of underlying models (Gruber, Vemuri & Rice, 1997). This is discussed in more detail in Section 8.5.

In the next section, I detail more thoroughly the functional, technical and user interface requirements for APECKS and discuss the ontology development lifecycle supported by the system. In Section 5.4, I describe how APECKS meets these requirements.

5.3 SYSTEM REQUIREMENTS

This section describes the requirements of a system design to support the collaborative construction of Living Ontologies. Section 5.3.1 describes the assumptions made about what is required from a system supporting collaborative work, the users of such a system and the context of its use. Section 5.3.2 outlines the usage lifecycle imagined for the system and Sections 5.3.3 and 5.3.4 discuss the functional and technical requirements for the system.

5.3.1 ASSUMPTIONS

This work combines the two strands of distributed collaborative work and ontological engineering. I make two fundamental assumptions arising from a constructivist view of ontological engineering as discussed in Section 2.4:

- Collaborative work involves reaching a shared understanding of a domain, or at least exploring the viewpoints of others in a group. In other words, collaborative work can be thought of as an ontological engineering task: the exploration of the design space of conceptualisations of a domain.
- Ontological engineering involves charting or mapping a design space of knowledge representations so that an appropriate one can be selected for use in a KBS. This process is an ongoing collaboration between knowledge engineers and domain experts. In other words, ontological engineering can be thought of as collaborative work.

The constructivist perspective emphasises two qualities of ontologies that are not recognised in the classical perspective, namely:

- Ontologies are only consensual, if at all, for the group that constructed them. This leads to the concept of *roles*: sharable knowledge representations that are explicitly not consensual. Roles are discussed in Section 5.3.1.1.
- Ontologies are mutable and alter over time as understanding of the domain modifies and new input is received. This leads to an emphasis on *change*, which is discussed in Section 5.3.1.2.

5.3.1.1 Roles in APECKS

Existing ontology servers hold ontologies that can only be changed by those to whom permission has explicitly been given. Although it may be possible for an ontology server to hold distinct ontologies on the same domain, there are no mechanisms for recognising that they address the same topic, nor for discovering any similarities or differences between them. However, different people do have distinct and different conceptualisations of a domain: the separate knowledge representations based on them are termed here *roles*.

I use the term 'role' instead of the term 'ontology' to mean a sharable knowledge representation of a particular domain. I avoid the word 'ontology' in this context because of the common association between ontologies and consensual knowledge representations (see Section 2.3). Under the constructivist viewpoint, the distinction between consensual and non-consensual knowledge representations is a false one since every ontology is only consensual across those who have constructed it (and even then, not necessarily across them). However, to make this distinction clear, I use 'role' to mean an ontology which has been created by a limited number of people, and 'ontology' to mean one which has been created by a larger number.

Other terms, such as 'viewpoint' or 'perspective' could be used instead of 'role'. In part, these terms were not chosen because they imply, to a certain extent, the same underlying representation but with parts hidden or emphasised according to the user. Roles, on the other hand, are based upon different representations entirely. 'Role' was also chosen because of the implication that the same user may have different roles. In APECKS as described here and used in the next two chapters, users generally construct only a single role each, and users cannot directly edit another's role. However, this is not necessarily the case. Users could construct different roles in which they consider the domain in the context of different tasks. Alternatively, different roles could focus on different areas within the domain. Equally, roles could be constructed collaboratively by multiple users. These possibilities are considered in the requirements for the system and discussed in more depth in Section 8.4.2.

5.3.1.2 Change in APECKS

As well as restricting who can make changes to an ontology, existing ontology servers give preference to the existing state of affairs, with changes having to satisfy certain constraints, such as consistency with the rest of the representation or the agreement of all the other collaborators (see Section 2.6.2.2). This may lead to stagnant and restricted ontologies.

Within the constructivist viewpoint, the emphasis is on an exploration of the design space of ontologies. In order to investigate different ontology structures easily, changes must be simple to do and undo. In addition, the analysis of the design space involves not only exploration in order to discover different options, but also evaluation of those options. When something is changed,

therefore, it is important to maintain a record of the reasons behind the change: why the new state is better than the old state.

5.3.1.3 APECKS in Use

The following assumptions are made as to the way APECKS will be used.

- **Users**

It is assumed that users will be distributed over several locations, possibly world wide, but will have access to the Internet through their computers. Their computer systems are assumed to be, at a minimum, able to run a simple text-based web browser such as Lynx.

Secondly, it is assumed that most users will not be familiar with knowledge engineering methodologies or terminology: they are domain experts rather than knowledge engineers. However, it is also assumed that some users will be knowledge engineers, and will require 'power features' in order to construct sophisticated ontologies.

Finally, it is assumed that users may not be limited to a particular collaborating group, and that those using the system may change over time. In other words, the users may be separated temporally as well as spatially.

- **Context**

The APECKS system is conceived as being a general system, which could be used in a number of situations and contexts, rather than a specific one tied to one particular domain. This means that it cannot be incorporated into users' normal workflow in the same way as, for example, JANUS (kitchens: Fischer et al., 1991) or NETWORK-HYDRA (computer networks: Fischer et al., 1992).

5.3.2 USAGE LIFECYCLE

The simple methodology supported by APECKS is illustrated in Figure 5.1. The methodology suggests a lifecycle involving four processes:

1. Seeding APECKS with information about the domain.
2. Constructing separate sharable knowledge representations, which I term *roles*.
3. Comparing roles to identify similarities and differences between them.
4. Discussing the roles and the results of comparisons between them.

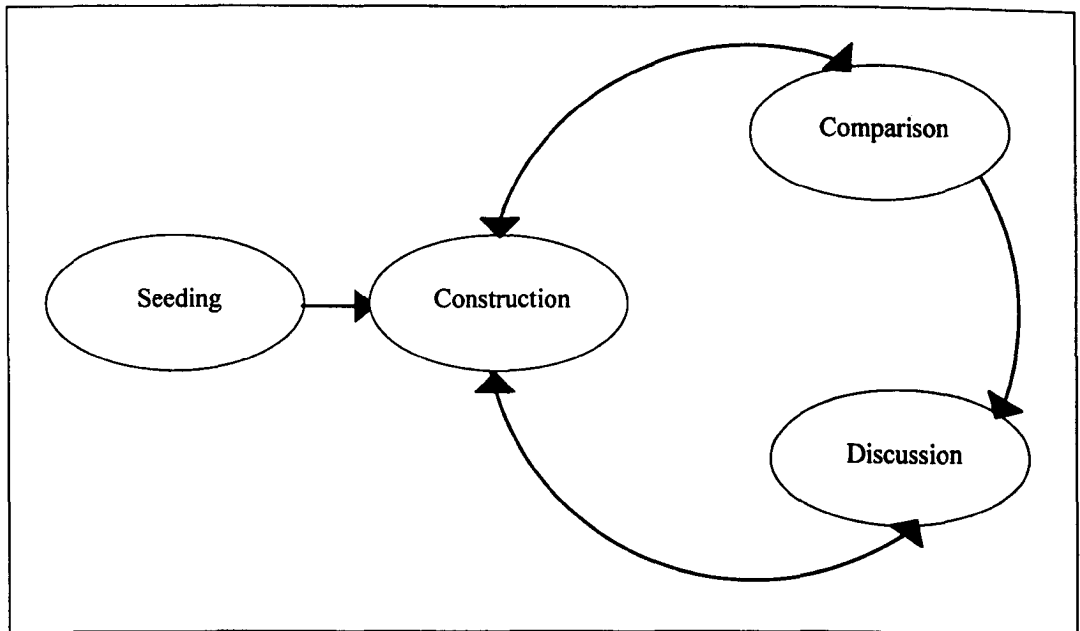


Figure 5.1: The APECKS lifecycle.

After the seeding of the domain, the lifecycle is essentially cyclical. The construction of roles leads to their comparison, which in turn sparks discussion about the domain, which may lead to reconstruction or alteration of roles. However, comparisons could lead to the alteration of roles without discussion and roles could be discussed independently, without comparing them with others.

5.3.3 FUNCTIONAL REQUIREMENTS

Below, I discuss each of the functional requirements of APECKS in order to support the four processes that constitute the lifecycle as detailed above. These requirements have been generated for a collaborative ontology server using the methodology outlined above: they may not be appropriate for ontology servers supporting other methodologies. Equally, they are not intended to dictate particular design decisions, although their fulfilment is a means by which ontology servers supporting this methodology can be assessed.

5.3.3.1 Seeding

The seeding process is a short stage during which the system is set up with information to start the collaborative process.

The first step in the methodology involves constructing user identities for those who will be interacting with the system. Persistent user identity is important within collaborative systems so that users know to whom they are talking (see Section 3.4.2.4). Additional information about users, such as their contact details and a reference to a home page on the WWW, provides status information which may aid communication. This generates the requirements:

1. Users should have persistent identities
2. User information should be stored and accessible to other users

The second step involves specifying the domain in enough detail that users can understand what it is that they are discussing and creating ontologies for. The kind of information that should be added at this stage is:

- An informal description of the domain.
- A number of items within the domain around which roles can be constructed. The items specified should be known or understandable to all those who will be contributing even if they think about them in different ways. For example, a geologist and a mason may have different conceptualisations of what granite is in terms of its classification while still agreeing that it is a rock that looks like a particular picture.

The kinds of information that should be specified about each item at this stage are those which would be accepted by all those contributing to the domain, for example its name, a description and a picture of the item. These form a consensual basis for discussion.

- Criteria against which roles that are generated can be judged. In Design Space Analysis (see Section 3.5.4), criteria are used to judge different options generated as responses to questions. Within APECKS, there is a requirement for a similar system of evaluating roles (roles can be conceived of as options generated as a response to the question "How is the domain structured?").

The items and criteria generated during the seeding phase specify the expected scope of the domain. The items chosen scope the breadth of information that will be structured, while the criteria may scope the types of tasks that are considered in the role construction. However, more items and criteria may be generated during the rest of the system lifecycle, widening the scope beyond that originally conceived by the system seeder. In particular, criteria may be generated as a result of the comparison and discussion of different roles and used to distinguish between them. Section 5.3.3.4 below discusses the use of criteria within APECKS.

To summarise, there is the following requirement:

3. Information about domains should be stored explicitly

5.3.3.2 Role Construction

The construction and maintenance of roles is the central activity within the methodology, both for knowledge engineers and for capturing knowledge from domain experts. As with all ontology servers (see Section 2.6.1.1), this requires the following:

4. Knowledge should be represented internally in a semi-formal knowledge representation
5. The internal knowledge representation should be transformable into a human-readable presentation
6. Users should be able to browse through the available knowledge
7. Users should be able to construct and edit knowledge representations

In addition, since APECKS needs to support both multiple domains and multiple roles within each of the domains, this gives the requirement:

8. Roles should be considered separately but retain a link to other roles within the same domain

Existing ontology servers are designed to give support to **knowledge engineers** in the construction of ontologies. As discussed above, APECKS is also geared towards the support of **domain experts**, who are not necessarily experts in the construction of ontologies. Therefore, as well as support for the direct manipulation of knowledge representations, APECKS needs to support direct, automated knowledge acquisition from domain experts (see Section 2.5.2.2). Direct knowledge acquisition involves contrived KA techniques (see Section 2.5.1), as these can be easily automated and analysed without the intervention of a knowledge engineer. This adds requirements for support that:

9. Users should be led step-by-step through knowledge acquisition process
10. The system should support contrived knowledge acquisition techniques for direct knowledge acquisition from users

5.3.3.3 Role Comparison

In order to identify similarities and differences between roles, to encourage reformulation, discussion and evaluation of the roles, they need to be compared. As knowledge engineers may not be involved in interactions with APECKS, there is a requirement that:

11. Knowledge representations should be compared in an automated fashion

Once comparisons have been made, they should be used to prompt reorganisation or reformulation of the roles, as well as their discussion and evaluation in terms of the criteria applied to the domain. There is therefore the requirement that:

12. Prompts should be generated for users to make changes to and discuss roles on the basis of comparisons

5.3.3.4 Discussion

As mentioned in Section 5.3.3.1 above, the methodology supported by the system is a Design Space Analysis of the possible conceptualisations of a domain. In terms of a simple QOC structure (see Section 3.5.4), the Question is "How should the domain be represented?" and the Options are the various roles that are generated. This brings two requirements:

13. Evaluative criteria should be represented within the system and viewable by users
14. Arguments concerning the validity of criteria and the applicability of criteria to roles should be represented within the system and viewable by users

The use of a specific formalism for discussion was touched on briefly in Section 3.6. The review of design rationale formalisms in Section 3.5 indicates that there is some, though not much, agreement between formalisms. At this stage, it is not possible to predict the type of speech acts that will be necessary in the construction and maintenance of ontologies: therefore no one formalism should be enforced. Instead, arguments can serve as general annotations about roles and the way in which they are structured, and thus fulfil the need for asynchronous communication between participants. However, asynchronous communication is sometimes not sufficient, so there is also a need for:

15. Users should be able to communicate synchronously

With all kinds of communication, their integration into the rest of the system is essential, and they should be stored for future use. There is thus a requirement that:

16. Users should be able to indicate areas under discussion directly within communication
17. Discussion between users should be archived for future use
18. Discussion archives should be searchable

5.3.3.5 Change Tracking

The demands on a change management system are dependent on the type of changes that can be made within it. For example, Microsoft® Word's change tracking system, formatting changes as well as the simple insertion and deletion of text, are recorded and can be searched within a single document. With hypermedia, where information is structured in a hypertext, it is harder for the user to identify where changes have been made. Chronological awareness tools have been

developed which generate information about changes in hypertext structures (Chen & Gaines, 1996) which can be client-side or server-side. Server-side chronological awareness tools such as CHRONO (Chen, 1996) keep track of changes to documents within a file structure on a server and create a page listing these changes. Similarly, commercial CASE tools and separate version-management software attempt to support change tracking from information in other formats.

As indicated in Section 5.3.1.2 above, change is very important within evolving systems like APECKS. Support is needed for recording the changes, recording the reasons for the changes and making the rationale available and explicit to users.

19. Changes to roles and to discussion threads should be recorded
20. Users should be able to construct a rationale explaining the changes made
21. Users should be able to browse, discuss and search the rationale

5.3.4 TECHNICAL REQUIREMENTS

- **Object-Oriented Database**

Frame-based knowledge representations (see Section 2.7) are object-oriented systems, and hypertext can be easily generated from them. In order to represent knowledge structures efficiently, the system should use an object-oriented database.

- **Internet Access**

Distributed collaborative work involves the collaboration of users from a number of spatially and temporally separated locations. The Internet and Intranets give the facility for communication between the different spatial locations. The system should be accessible through the Internet or within Intranets.

- **Use of Networked Resources**

There are several ongoing knowledge engineering and knowledge acquisition projects going on over the WWW, providing access to existing ontologies, and knowledge acquisition and knowledge analysis tools (see Chapter 2 for some examples). APECKS should be able to utilise these other facilities as a client, while also making its own resources available as a server.

- **Mirroring**

While the Internet makes information available to any connected computer in the world, network delays mean that, practically, it is difficult to access some sites simply due to the time it takes to get information from them. With highly interactive sites, such as applications made available over the WWW, time delays are even more frustrating as they take up a

greater percentage of the time. There should therefore be a way of mirroring information available on APECKS, so that users can interact with their closest site while still interacting with users on the other side of the world.

5.4 SYSTEM DESCRIPTION

The previous section outlined the requirements for the APECKS system. This section details its implementation.

5.4.1 ARCHITECTURE

As with other Ontology servers, APECKS satisfies the requirement for distributed access to a hypertext structure by using the World Wide Web (WWW). APECKS is based on an internet-accessible multi-user text-based virtual environment called a MOO (Multi-user domain - object-oriented; Curtis, 1992; see Section 3.4). MOOs can run on any Unix machine or under Windows NT, and match the following requirements as an implementation platform for the APECKS system:

- **Object oriented database**

MOOs consist of two parts: an object-oriented database, which defines each individual MOO, and a server program, which interprets and runs the database file. The object-oriented nature of MOOs means that object-oriented representations, like the frame representations used in knowledge engineering, are very simple to employ.

- **Internet accessibility**

MOOs are accessible via the Internet using the telnet protocol. The mechanisms for opening, closing and maintaining connections between computers necessary for ontology servers are built into the MOO server program. Several MOO developers have built on top of this and made available MOO databases that allow objects within them to be viewed and manipulated through the WWW.

- **Programmability and rapid prototyping**

MOOs are programming environments. Any object within a MOO database can be programmed to interact with users or other objects. Because these programs are part of the database itself, they do not have to be compiled, leading to rapid prototyping. For the purposes of behaving as a WWW server, MOOs can be programmed to generate HTML dynamically.

- **Multi-user synchronous communication**

MOOs allow multiple users to connect to them and interact with each other using synchronous text-based communication. Built into the standard LambdaMOO database (Curtis, 1992) are persistent user representations, which facilitate user modelling, and other communication methods, such as asynchronous mail.

As well as being a server of ontologies, APECKS needs to act as a client in order to access network-accessible knowledge acquisition applications and other ontology servers. To fulfil these requirements, three modifications were made to the standard LambdaMOO database (Curtis, 1992).

- **HTTP server and client**

The APECKS database understands HTTP1.1, and can therefore act as both a WWW server and, when in the role of a proxy or gateway, a WWW client. The standard HTTP Basic Authentication system is used to establish user identities within interactions.

- **Dynamic generation of HTML**

The APECKS database generates HTML dynamically on the basis of its internal object representations, enabling the user to interact with the objects using a WWW browser.

Applications that dynamically generate HTML pages often generate the entire page internally before sending the page to the client. This both delays the initial receipt of information by the user and extends the time taken for each transaction. Within APECKS, the structure of a page and its content are generated separately. The structure determines the sections that need to be generated and the order in which they are sent to the user, but the sections themselves are generated in parallel. This speeds up the construction of pages, and has two other advantages. Firstly, those adapting APECKS can also easily change the structures of pages without touching the content, or vice versa. Secondly, either the structure or the content of the page, or both, can be determined by the user identity and preferences, facilitating the adaptive presentation of information.

- **Mirrored access**

The APECKS database includes facilities for setting up mirrors in other locations around the world. The mirrors hold the same information as each other, but can be accessed faster by those nearer to them. Activity by users at one site can be seen by those at mirroring sites, so that users can transparently communicate, both synchronously and asynchronously, across sites.

The modified database developed during the course of this research has been used in electronic conferences (Hardy et al., 1997a; Hardy et al., 1997b; Hardy et al., 1998) and within a

collaborative learning environment (Gibbs, 1998). These applications did not use any of the advanced ontology management features that are built into APECKS: the integration of a MOO with the World Wide Web is a useful application in its own right.

5.4.2 KNOWLEDGE REPRESENTATION

APECKS is a Frame Representation System that represents ontologies in a manner based on that defined by the Frame Ontology (Karp & Gruber, 1997) and described in Section 2.7. The use of this representation ontology means that the internal representations are easily translated into the Generic Frame Protocol (GFP) or KIF (see Section 2.6.1). GFP and KIF are designed for communication between knowledge based applications. By using the same ontological commitments, translation between APECKS and Ontolingua (and other applications using the Frame Ontology) should be made easier. This enables the seamless incorporation of ontologies from other ontology servers into APECKS (see Section 8.3.3).

The basis of the APECKS knowledge representation is a number of **individuals**, which are concepts within the domain that are not classes. Individuals are grouped into **classes** and have **slots**, which define **values** and have **facets**. APECKS handles multiple inheritance through the class hierarchy, such that the value of a slot for an individual can be inherited from its types (classes). Slots can hold many different kinds of values, and can hold many at the same time.

Some restrictions apply within APECKS as the knowledge representation is only intended as a proof of concept: the class hierarchy is primarily determined by the membership of individuals to classes rather than being directly defined by the user; users cannot define facets or axioms. In the main, these restrictions serve to make it easier for users who are not knowledge engineers to understand and use the system.

Knowledge stored within an APECKS database is divided into a number of **domains**. Within each domain are defined a number of **roles**. There is some overlap between roles, in that the same individuals can be classified in many different roles. Each individual has slots for information which define it uniquely, such as its name and description, but the class membership of an individual, its inherited slots and the values of those slots, may vary from role to role. A representation of the similarities and differences between two or more roles is known as a **comparison**: these are discussed in detail in Section 5.4.5, on comparisons of ontologies within APECKS.

On top of those objects that encode the ontologies, there are objects that facilitate discussion about them. There are two types of these objects: **criteria**, which are short, general, evaluative assertions about the reasons for a particular state of affairs; and **annotations**, which are longer discussions, sometimes with more than one author. These are discussed in detail in Section 5.4.6, on communication within APECKS.

APECKS has to supply knowledge both to those using the system through the WWW and to other knowledge-based applications. In order to supply the information, however, APECKS must first convert it from the internal representation described here into an external representation that can be transmitted over the Internet and understood by the receiver. Three examples of these external representations are:

- **HTML** for display within users' web browsers (see Section 5.4.3)
- **HTML form submissions** when passing information to WebGrid-II (see Section 5.4.4.2 and Section 5.4.5)
- **KIF** (the Knowledge Interchange Format) when exchanging information with other ontology servers (see Section 8.3.3).

Each of these formats requires a separate generation process and, within most, some of the information APECKS stores internally is lost. This is especially clear in the translation to HTML, where all information about the purpose of the transmitted text is lost: the purpose of text within HTML pages is only indicated implicitly by its formatting.

5.4.3 USER INTERFACE

The user interface design was limited by the requirement for APECKS to be usable with the simplest of web browsers currently available on low-end computers (see Section 5.3.1.3). Designing for the lowest common denominator means that the interface was limited by the interactions available within HTML, which are outlined in Section 4.3.2.

5.4.3.1 Connecting to APECKS

Users connect to APECKS by entering the URL of one of the pages within the system into a web browser. There is no single entry-point through which users must pass, although the URL [http://hostname:port/\\$domain/domains](http://hostname:port/$domain/domains) is a good general starting point as it displays a list of all the domains within the system with descriptions. Having no single entry point means that it is easy for users to exchange URLs within emails etc. When they connect, users are prompted for their username and password that identifies them uniquely within the system. Users without a username can connect as 'guest' with no password.

5.4.3.2 Objects and Pages

Individual WWW pages within APECKS are identified as a combination of an object and a particular view of that object (a page). URLs are of the form:

<http://hostname:port/object/page>

In the case of the knowledge representations stored within APECKS, the objects are individuals, classes, slots, roles, domains, comparisons, criteria or annotations, as described in Section 5.4.2 above. Each object has a number of pages or views associated with it.

The 'View' page of the object gives essential information about it, including its name and description and other relevant information depending on the type of object, such as slot values, subclasses and slot constraints. These characteristics can be edited using HTML forms within the 'Edit' page.

The 'Changes' view gives a list of all the changes that have happened to the object since the list of changes was last looked at. This page also offers a search engine, allowing a user to search for changes on an object by date, the person who made the change, and other details about the change.

The 'Comments' view gives a list of the annotations and criteria that pertain to the object. This page also offers a search engine, allowing a user to search for comments on an object by date, the person who made the comment, for keywords within the comment and other objects to which the comment is related.

Finally, to support use by domain experts as well as knowledge engineers, context sensitive help is available for every type of object. For roles and classes, users may also view pages which display list of suggested actions, supporting knowledge acquisition from the user. Separate pages also give links to views of related objects within other roles.

5.4.3.3 Interaction Mechanisms

There are several distinct forms of interaction between the user and APECKS. These are limited to those available within HTML 3.2 (World Wide Web Consortium, 1997), namely to simple HTML links and forms.

Links

Standard links from text within pages usually link the name of an object (be it a role, class, individual or whatever) to the main view of that object, thus allowing navigation between objects and through the knowledge representation. In almost every case that the name of an object appears, it will be given as a link to that object, allowing immediate navigation to it. The exception to this is within the main view of a comparison, in which the possible actions listed act as links to complete the action.

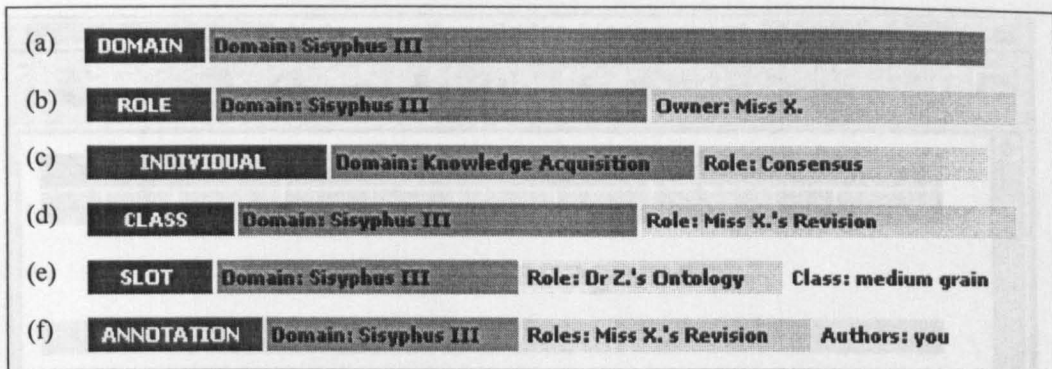


Figure 5.2: Status bars from (a) a domain, (b) a role, (c) an individual, (d) a class, (e) a slot and (f) an annotation. The individual is being viewed under the 'Consensus' role and the slot is currently focusing on values for the class 'medium grain'.

Links are also used within the status bar at the top of every page to give links to *landmark* objects associated with the object being viewed (see Figure 5.2). Links to these landmark objects reduce cognitive overhead and disorientation (see Section 4.3.1) by allowing users to navigate back to familiar, central pages. The landmark objects that are accessible depend on the type of object being looked at, but generally include the domain and role to which the object belongs. For individuals, the status bar indicates the role under which the individual is currently being viewed, since an individual can hold different information under different roles. For slots, the status bar indicates the class for which values are currently being viewed or edited. Discussion objects and roles also indicate their owner within the status bar, so that information about the author of the object can be directly accessed.

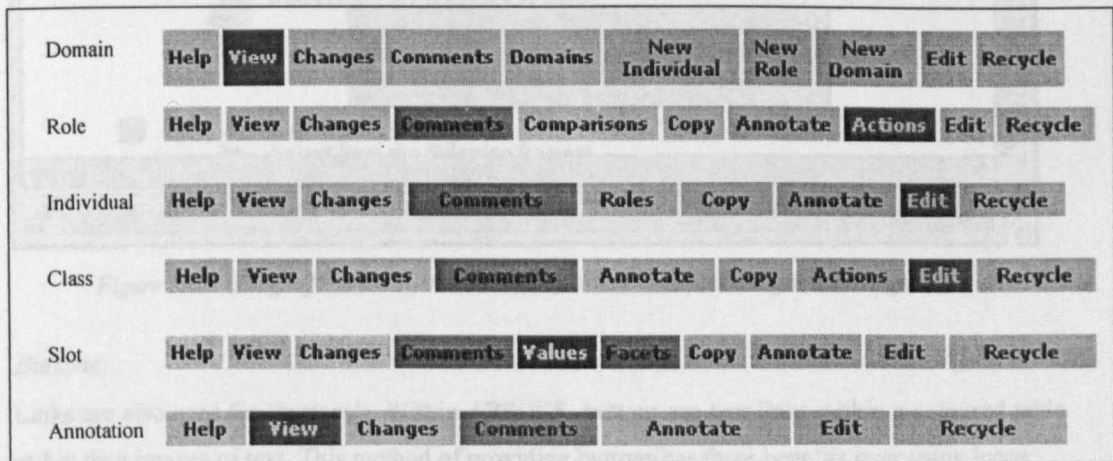


Figure 5.3: Button bars from a domain, role, individual, class, slot and annotation.

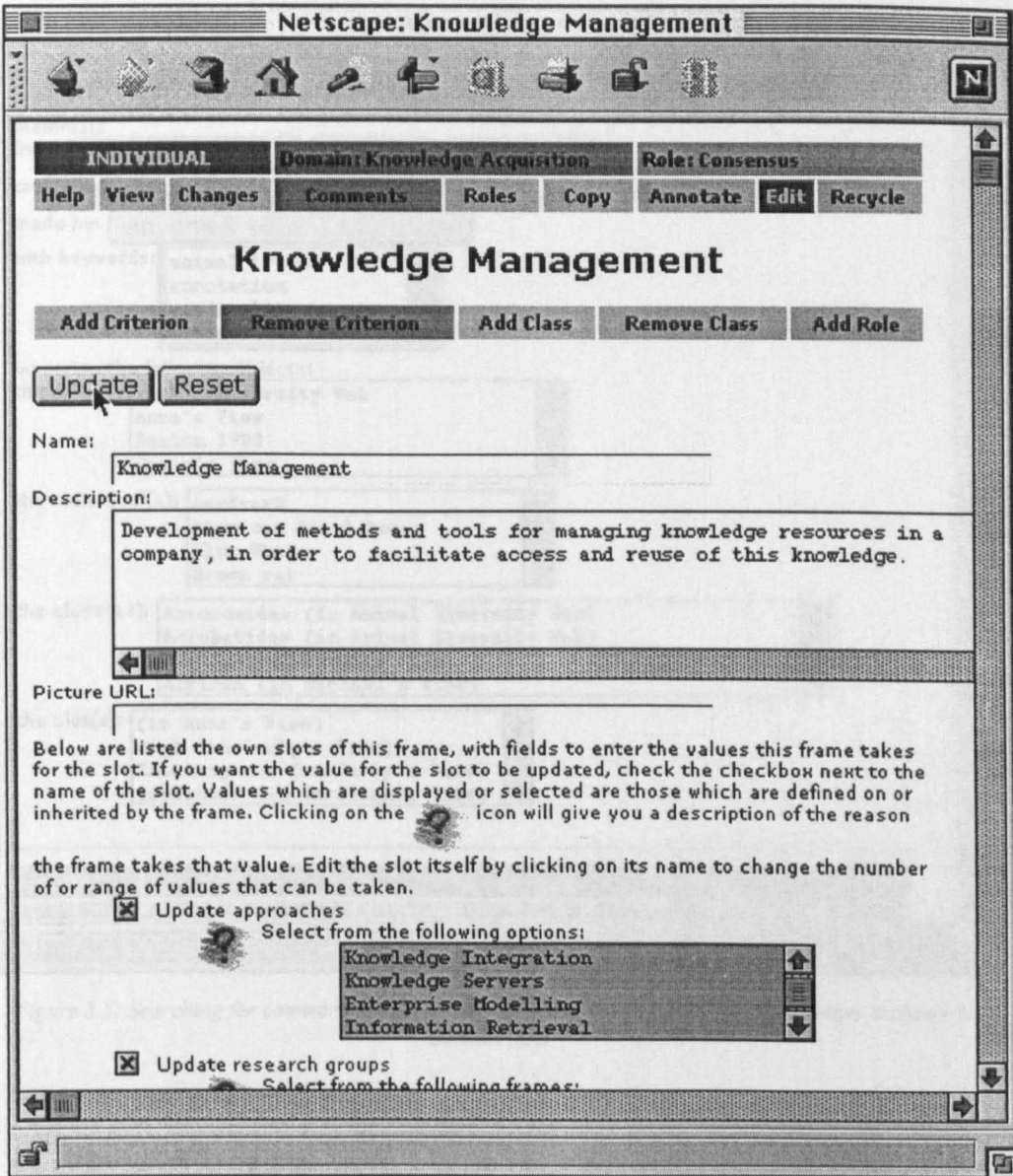


Figure 5.4: Editing information on an individual. More examples are given in Chapter 6.

Buttons

Links are also used for 'buttons'. Within APECKS, buttons are text links within a coloured table rather than images of text. This method of providing buttons has three benefits over using icons:

- **extensibility**, as new images do not have to be created for a new button
- **decreased download times**, as images do not have to be loaded separately
- **accessibility** no matter which web browser is used (although older web browsers will not format the buttons as a table)

Forms for Changing Information

search_comments Search for annotations and criteria on

Mammals

from: January 15 1998

to: June 24 1998

made by: anyone

with keywords: animal types
annotation
cardinality
categorical

involving the following objects:
the role(s): Animal Diversity Web
Anna's View
Benton 1990
Carly's View

the individual(s): Aardvark
American black bear
Blue Whale
Brown rat

the class(es): Abrocomidae (in Animal Diversity Web)
Acrobatidae (in Animal Diversity Web)
Africa (in Oliver's Role)
African (in Michael's View)

the slot(s): (in Anna's View)
diet (in Carly's View)
Intelligence (in Carly's View)
predators (in Caroline's View)

DOMAIN Domain: Mammals

Figure 5.5: Searching for comments within APECKS. The form for searching for changes is shown in Section 6.5.7.

Buttons are used in two main locations:

- at the top of every page within a button bar, which gives access to different views of the object (see Figure 5.3). The highlighted button, with a dark background, shows the current page the user is looking at. Shaded out buttons indicate areas that are not accessible or do not contain any information (such as the 'comments' page when no annotations have been made on the object).
- near the top of 'edit' pages, giving actions involving adding or removing other objects from it.

Icons

Icons are used to indicate whether changes have been annotated and to provide links to annotations, creating annotations, accepting prompts to carry out actions and requesting more information about slot values.

Forms for Changing Information

HTML forms are used in order to edit the knowledge representation within the 'Edit' page and subsidiary pages of each object. They are also used to obtain information during knowledge acquisition. When forms are used in this way, the entire page is a form (see Figure 5.4).

Forms for Searching

Forms are also used to search the archive of changes made to objects and comments and criteria made on them. When forms are used in this way, they appear at the bottom of the 'Comments' and 'Changes' pages (see Figure 5.5).

5.4.4 CONSTRUCTION SUPPORT

The main task performed by users, especially when they first start using the system, is to build a representation of their conception of the domain. Users first construct a role object to hold their conceptualisation. Within that role, they are then free to classify any or all of the individuals within the domain according to their own view.

For use within this section, I will take the example of a simple ontology in the domain 'People'. The individuals that users would start with might be famous people, or people within the organisation in which they worked. Part of the ontology is shown in Figure 5.6. The solid ovals represent classes and the solid rectangles slots, while the dashed ovals indicate individuals and the dashed rectangles subclass partitions. Thus, the class 'people' has two slots defined on it: 'age' and 'gender'. 'People' has two subclasses, 'children' and 'adults', which make up a subclass partition. 'Children' has two subclasses, 'infant' and 'teenager', which make up a subclass partition, and 'adults' has three subclasses, 'young', 'middle-aged' and 'elderly', which also make up a subclass partition. Each of the classes at the bottom of the hierarchy has three instances: the ontology is based on 15 people in total.

APECKS supports the process of ontology construction for knowledge engineers by allowing them to create directly new individuals, classes, and slots and assigning properties to them (see Section 6.2 for examples). However, because APECKS is also intended for use by people who are not experts at the construction of ontologies, extra knowledge acquisition support is given both internally and externally, in the form of WebGrid (Gaines & Shaw, 1996a & 1996b: see Section 6.3.7 for examples that use WebGrid).

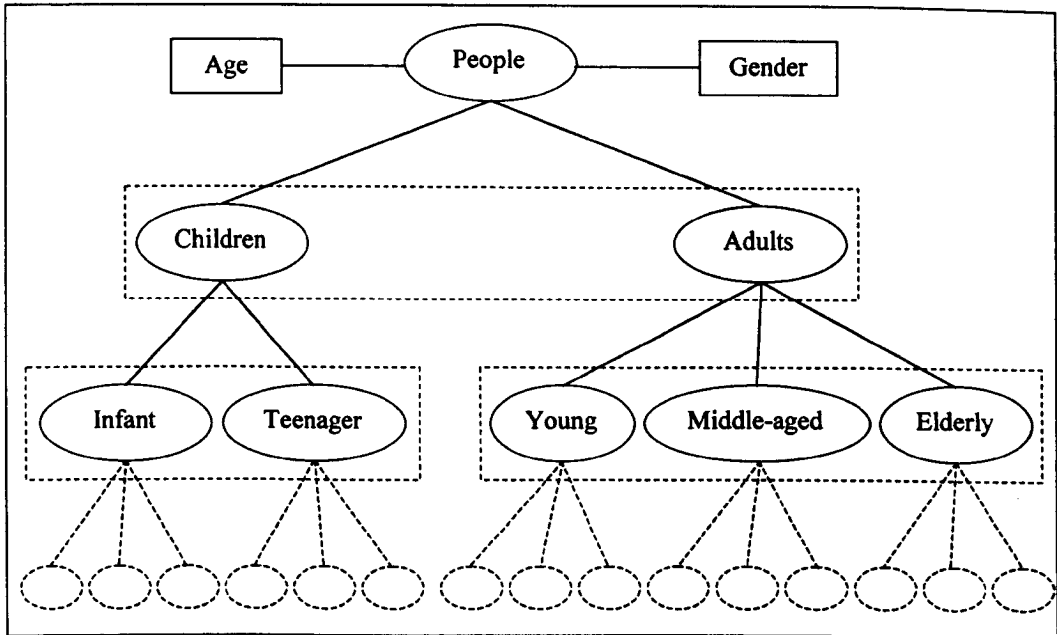


Figure 5.6: The 'People' ontology.

5.4.4.1 Internal Knowledge Acquisition Support

Knowledge acquisition support is given within APECKS using pages listing possible actions that can be carried out by the user. The listed actions prompt the user into making changes of three kinds:

- **expand** the role by adding individuals or classes (see Section 6.3.4 for an example). This takes the place of a Laddered Grid (see Section 2.5.1.1) by expanding the coverage and exploring the structure of the user's conceptualisation of the domain. For large classes, with lots of direct instances, APECKS prompts the user to create subclasses. For small classes, with no or few instances, APECKS prompts the user to create examples of the class.
- **rerepresent** parts of the role, such as by rerepresenting categorical slots that cannot take more than one value as subclass partitions and vice versa (see Section 6.3.11 for an example). These are convenience methods: categorical slots and subclass partitions perform the same function of dividing the instances of a class into mutually exclusive sets. The choice between them is left to the builder of the role. If the minimum slot cardinality is also one (i.e. individuals **have** to take a value for the slot), the subclass partition formed is exhaustive, and vice versa.

Using the people ontology given in Figure 5.6 as an example, the subclass partition of 'adults', which includes the subclasses 'young', 'middle-aged' and 'elderly', could be converted into a categorical slot. The slot, which would be defined on the class 'adults', might be named 'maturity' and would have the possible values 'young', 'middle-aged' or

'elderly'. Those previously in the class 'young' would take the value 'young', those in 'middle-aged', the value 'middle-aged' and so on.

Similarly, the slot 'gender', which has the possible values 'male' and 'female', could be converted into a subclass partition. The subclass partition, which would be given for the class 'people', would include the classes 'male' and 'female'. Those individuals that took the value 'male' would be in the class 'male', while those who took the value 'female' would be assigned to the class 'female'. Since all individuals would have to take one or other of the values, the subclass partition would be exhaustive.

- **maintain consistency** when the role is inconsistent (see Section 6.3.10 for an example). Ontologies constructed within APECKS may eventually be used within a KBS, and therefore need to be consistent within themselves. On the other hand, enforcing consistency forces users to formalise early on in the brainstorming phase of ontology construction: it is useful to allow inconsistent ontologies during this phase. Prompts for maintaining consistency help to bridge the gap. They also indicate areas that may be incorrect by pointing out the reasons behind the inconsistencies. Section 5.4.4.3 details the extent to which consistency is checked and the areas in which it is enforced.

Users are free to choose which, if any, of the actions to take next. For any of these action prompts, the user is also given the opportunity to create an annotation explaining the current state of affairs (see Section 6.3.6 for an example). Making such an annotation prevents the action prompt being shown again so that users can discard prompts which are not useful. These annotations become part of the design rationale for the ontology as they contain information regarding choices that were made during its construction.

The prompts for expansion of the role are directed towards areas that are under-developed. For example, users are prompted to create more examples of classes that have few individuals, and to create subclasses of large classes. The creation of subclasses involves a card sort (see Section 2.5.1.2), limited by the HTML form interface, involving the user identifying the piles before dividing the class instances between them (see Sections 6.3.4 and 6.3.5 for an example).

5.4.4.2 External Knowledge Acquisition Support: WebGrid

Repertory grids (see Section 2.5.1.3) are one of the more powerful knowledge acquisition techniques available to knowledge engineers, particularly as they are easy to automate. APECKS uses the network-accessible WebGrid-II server (Gaines & Shaw, 1998) to provide this technique for its users. WebGrid-II acts as an intermediary between HTTP clients (usually WWW browsers) and Repertory Grid elicitation, analysis, comparison, modelling and inference tools, enabling these tools to be accessed through the WWW.

	Alice	Bill	Clare	David	Emma	Fred	Gemma	Harry	Imogen	Jon	Kate	Lionel	Mary	Nigel	Olwyn
Age	8	27	17	63	14	3	42	5	23	33	50	16	77	46	82
Gender	f	m	f	m	f	m	f	m	f	m	f	m	f	m	f
People	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
People:adulthood	ch	ad	ch	ad	ch	ch	ad	ch	ad	ad	ad	ch	ad	ad	ad
Children:maturity	in	?	te	?	te	in	?	in	?	?	?	te	?	?	?
Adults:maturity	?	yo	?	el	?	?	mi	?	yo	yo	mi	?	el	mi	el

Figure 5.7: The repertory grid arising from the first translation of the 'people' ontology.

WebGrid was not originally designed to be used by external applications in the way APECKS uses it. It does not make the elicited knowledge available in a standard external knowledge representation such as KIF, but instead uses a special encoding within HTML pages. The challenge of incorporating WebGrid into APECKS without altering WebGrid's normal operation also tested WebGrid's design and expanded its use.

During a user's interaction with WebGrid, APECKS acts as a gateway: it behaves like an HTTP client towards WebGrid-II, while still acting as an HTTP server for the user's WWW browser. Throughout the interaction, the user is presented with the same navigation bar as is normally viewed in APECKS, so that the process is as transparent as possible (see Section 6.3.7 for an example). APECKS performs three processes that enable this facility: the translation of the ontologies into grids; behaving as a gateway for requests to WebGrid; and the translation of grids into ontologies.

Translating APECKS ontologies to WebGrid grids

WebGrid encodes grids using hidden fields within HTML pages, which are submitted with each action the user takes. This client-side representation differs from APECKS's server-side representation.

To initiate an interaction with WebGrid, APECKS must first generate a form submission that, when submitted to WebGrid, will encode the knowledge representation of a role or subsection of a role. Thus the initial step is the translation from APECKS's internal knowledge representation to WebGrid's hidden fields, which is carried out once, at the beginning of an interaction between the user and WebGrid. Submission of these hidden fields results in the WebGrid's main page, which gives prompts for triad elicitation, the editing of constructs and elements and so on.

The major problem in the linkup between APECKS and WebGrid lies in the translation between the hierarchical structure used by APECKS and the flat structure used by WebGrid and back

again. APECKS stores **explicit** information about hierarchical structures, through the use of classes. Analysis techniques provided within WebGrid-II can expose hierarchical structures within grids (Shaw & Gaines, 1998) from ratings on constructs, but users may not initially supply these dimensions as slots. In addition, since APECKS users are also intended to be domain experts, they are not necessarily adept at using and interpreting the outcome of these grid analysis techniques, while the usually graphical output is not amenable to machine interpretation. My solution to this problem involves isolating three separate grids within each role, and offering users the facility of using WebGrid with any one of these grids at a time.

The simplest and most usual translation between APECKS and WebGrid involves each instance within a role is represented as an element within WebGrid and each slot and class is represented as a construct. Figure 5.7 shows the repertory grid generated through this process on the 'People' ontology defined in Figure 5.6. The values of the slots 'age' and 'gender' are directly translated into constructs within the repertory grid. The class 'people' is translated into a boolean construct that can take the values 1 (not-in-people) or 2 (in-people). The subclass partition 'adulthood' defined on the class 'people' is translated into a categorical construct called 'People:adulthood', which can take the values 'children' or 'adult'. The other subclass partitions are similarly translated, with '?' indicating values that are not given.

The second kind of grid that can be constructed involves the elements within WebGrid being based on classes within APECKS, the constructs being based on slots and the values taken by the constructs being the default values for the instances of the class. Figure 5.8 shows a repertory grid generated through this process on the 'People' ontology defined in Figure 5.6. Each of the classes within the ontology is translated into an element. Each of the slots is directly translated into a construct, with the values taken being the default value the classes take for the slot. Thus, the default value taken for the slot 'age' is '35' for the class 'people', but '12' for the class 'children', '6' for the class 'infant' and so on.

	People	Children	Infant	Teenager	Adults	Young	Middle-aged	Elderly
Age	35	12	6	16	50	28	45	70
Gender	m	m	m	m	m	m	m	f

Figure 5.8: The repertory grid arising from the second translation of the 'people' ontology.

	Inverse	Inherited	Symmetric	Asymmetric	Reflexive	Irreflexive	Transitive	Weakly transitive
Age	?	2	1	1	1	1	1	1
Gender	?	2	1	1	1	1	1	1
Siblings	Si	2	2	1	1	2	1	2
Children	Pa	2	1	2	1	2	1	1
Parents	Ch	2	1	2	1	2	1	1
Ancestors	De	2	1	2	1	2	2	1
Descendants	An	2	1	2	1	2	2	1
Family	Fa	2	2	1	2	1	2	1

Figure 5.9: The repertory grid arising from the third translation of the 'people' ontology.

The third, and least used of the grids that are created by APECKS is one that allows users to gain an overview of the types of slots that are used within a role. This allows users to set various characteristics of slots: the inverse, whether it is inherited, and its symmetricalness, reflexivity and transitivity. Figure 5.9 shows the repertory grid generated by this process for the 'People' ontology defined in Figure 5.6 (with a few extra slots added). The elements are various internal properties of slots (when facets are supported, this grid will become more useful as they will be included as elements: see Section 8.3.1.2). The constructs are the slots defined within the ontology, and the values are their values on the slot properties. The grid shows that the slot 'Siblings' is the inverse of itself, is symmetric, irreflexive and weakly transitive, whereas 'Ancestors' is the inverse of 'Descendants' and is asymmetric, irreflexive and transitive. The logical implications of these slot properties are discussed in Section 5.4.4.3.

HTML hidden fields are constructed which encode these grids on the pages that prompt for actions. When users submit these forms, the transaction with WebGrid begins, and APECKS begins the next process: acting as a gateway.

APECKS as a WWW Gateway

APECKS acts as a gateway while the user interacts with WebGrid, with all HTTP requests to and responses from WebGrid passing through APECKS. Requests are passed from the user's web browser exactly as-is, but the response goes through three processes before being sent to the user:

- **Logging of WebGrid fields**

The WebGrid fields that are contained within the response are logged.

- **Changing of URLs**

URLs that point directly to WebGrid are changed so that the transaction continues to take place using APECKS as a gateway.

- **Checking for invalid WebGrid fields**

The most important filtering of the page sent by WebGrid in response to the user request is to check for changes that have been made that, while valid within WebGrid, would cause problems for APECKS. The fields from the previous submission are checked to make sure that the user has not renamed elements, changed the way that class membership has been encoded, or made other alterations to the grid which would disrupt the translation from the WebGrid grid back to APECKS's representation. If such changes have been made, the user is warned and the response altered to undo the change.

Translating WebGrid grids to APECKS ontologies

When a user requests a page within a role that may have been altered on the basis of a WebGrid interaction, the recorded fields from the most recent response from WebGrid are used to make changes to the APECKS ontology. Changes to APECKS are postponed until then to reduce the amount of processing that APECKS must perform during a user's interaction with WebGrid. The user is not required to act to indicate the end of a WebGrid session, as this would compromise the aim of transparency between APECKS and WebGrid.

5.4.4.3 Consistency Checking

In general, APECKS allows users to build inconsistent, incomplete roles if they want to. As discussed in Section 2.6.1, inconsistent, incomplete or ambiguous knowledge level models are useful in some contexts, while consistent, complete and unambiguous ones are useful in others. Formalisation during the early stages of construction may force an ontology into a structure prematurely and make it difficult to make changes later. Ontology servers that enforce consistency, as discussed in Section 2.6.2.2, may suffer in this way.

Having said that, in later stages and for knowledge engineers hoping to use the ontologies they create within KBSs, consistency is important. As well as prompting users to make changes that will bring their role to a consistent state (see Section 5.4.4.1), APECKS does encourage consistency in three other ways: through subclass partitions, the class hierarchy and slot values.

Subclass partitions

Subclass partitions are defined as sets of subclasses of a common superclass that hold no instances in common with other subclasses in the set. In other words, the subclasses form an exclusive set.

For example, within the 'People' ontology shown in Figure 5.6, people in the class 'children' cannot be in the class 'adults'. Users can define subclass partitions in three ways:

- Directly when editing a class, either for a set of existing subclasses or for new ones (see Section 6.2.5 for an example)
- Through the internal knowledge acquisition support for card sorts (see Sections 6.3.4 and 6.3.5 for an example)
- From a categorical slot (see Section 6.3.11 for an example)

In each case, users have to state explicitly whether the partition is exhaustive (whether all instances of the superclass are instances of one or other of the subclasses) or not. For example, people must be either 'children' or 'adults': the subclass partition to which they belong is exhaustive.

Consistency is enforced once a partition is defined, by only allowing subclasses within the partition to have instances added to them that are not instances of any of the other subclasses in the partition. For example, the class 'children' can only have individuals added to it that are not already instances of the class 'adults'. If the partition is defined for existing classes, these classes may not in fact form an exclusive set: thus, it is possible to have inconsistent subclass partitions if desired. For example, when the partition is defined, there might be some people who have (either accidentally or deliberately) been classified as both 'children' and 'adults': this would be inconsistent with the definition of the partition.

The instances of the superclass and subclasses within a partition are checked for consistency, and inconsistencies are indicated both in the presentation of lists of instances and in the knowledge acquisition prompts. The inconsistent situations are:

- When an instance is in two or more of the classes in the subclass partition
- When the partition is defined as exhaustive, but some particular instance of the superclass is not an instance of any of the classes in the subclass partition
- When an instance of one of the subclasses in the partition is not an instance of the superclass

Class hierarchy

The class hierarchy is automatically generated based on the class membership of individuals. A subclass, by definition, holds a subset of the instances of the superclass. The class hierarchy is produced to fulfil this definition. In most cases, the only way the user can change the class hierarchy is by changing the classification of individuals. However, subclass partitions define superclass-subclass relationships explicitly, and thus override the computed hierarchy. Taking the role described in Figure 5.6 as an example: if the subclass partitions were not defined and a new

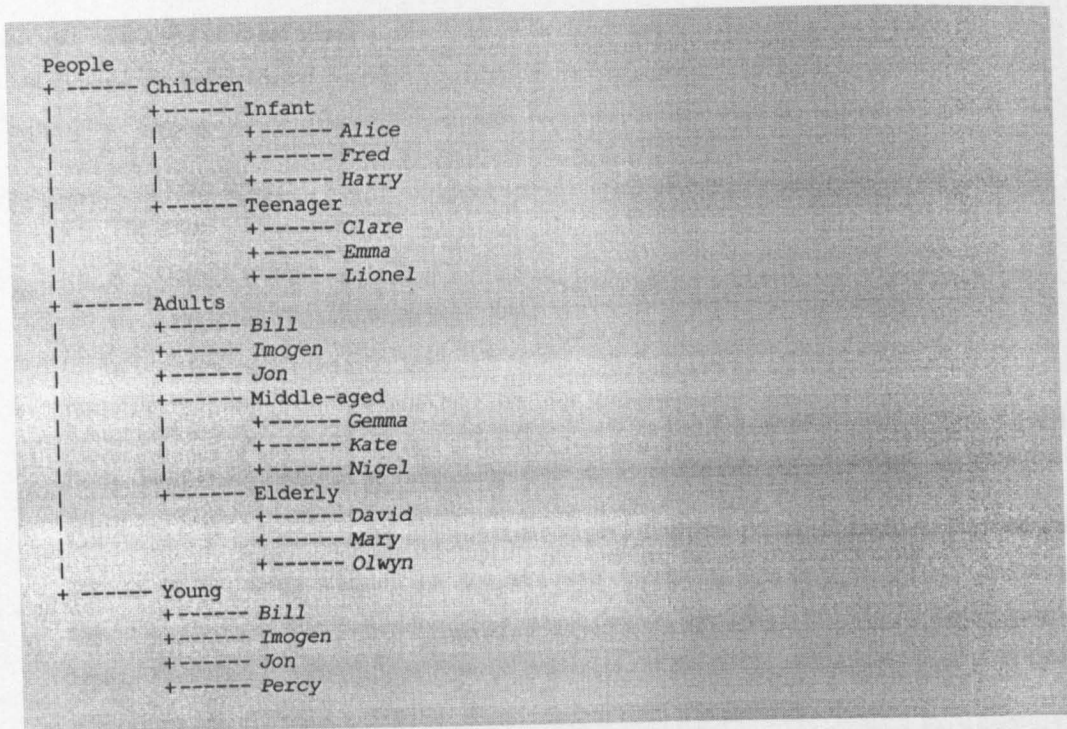


Figure 5.10: The hierarchy after the addition of 'Percy' to the class 'Young'. *Italics indicate individuals.*

individual, 'Percy' was added to the class 'young', but not the class 'adult', the hierarchy would alter. 'Young' could no longer logically be a subclass of 'adult', giving rise to the hierarchy shown in Figure 5.10.

Slot values

Slots define values for classes and individuals. The types of value that a slot can take are:

- Primitive values:
 - Numbers
 - Short strings (strings)
 - Long strings (texts)
- Links to individuals

A number of limitations can be applied to the value(s) that a slot can take for a particular class or individual:

- **Range**

Ranges of values can be defined on the slot itself, classes and individuals (see Section 6.3.9 for an example). The range can be limited in terms of any mixture of:

- The type of value taken
 - Strings
 - Texts
 - Numbers
- The actual values taken
 - Certain strings
 - Maximum and minimum numbers
 - Instances of certain classes

Ranges are inherited by classes from its superclasses or, if it has no superclasses, from the slot itself. Ranges are inherited by individuals from its types (the classes it is an instance of).

Consistency is enforced in the definition of ranges. Classes can only define ranges that are subsets of the range defined on the slot and subsets of the ranges defined on their superclasses. Individuals can only define ranges that are subsets of the range defined on their types. However, it is possible to have inconsistency if the range of a slot or class is defined **after** the range of a class or individual that inherits from it is defined.

For example, using the 'People' ontology, the range of the slot 'age' might be defined as a number, with a minimum of 0 and a maximum of 100. The range for the slot for the class 'adults' must be a subset of this range, so it would not be possible to extend the range to include the ages up to 150.

Consistency is also enforced on the values that can be taken by either directly by an individual or as default or 'must have' values by a class. Users can only assign values to individuals that are within the range defined on the individual (or inherited from its types). Similarly, users can only assign default or 'must have' values to a class that are within the range defined on the class (or inherited from its superclasses, or the slot itself). However, it is possible to have inconsistency if the range of a slot or class is defined after a value is assigned to a class or individual that inherits from it.

For example, using the 'People' ontology, the range of the slot 'age' for the class 'children' might be a number, with a minimum of 0 and a maximum of 19. The default values for the classes 'infant' and 'teenager' must be between 0 and 19. Similarly, the ages given for each of the instances of 'children' must be between these values.

The ranges and values are both checked for consistency, and inconsistencies are indicated to the user. These occur when:

- the range defined on the class or individual is not a subset of the range it inherits
- a value assigned to a class or individual is not within the range defined on or inherited by it

Users are given prompts for changing either the value or range that is inconsistent with the inherited range, and for changing the inherited range itself (see Section 6.3.10 for an example).

- **'Must have' values**

'Must have' values are defined on classes and indicate values that have to be taken by the instances of a class. Consistency is not enforced on this, but the values are indicated to users when they are assigning values to individuals. Where there is inconsistency, users are prompted to add the values that should be taken, or change the 'must have' values that are causing the inconsistencies.

For example, the 'People' ontology might define a class 'The Robinson Family' that had as instances all the people within that family. Those within that class, by definition, have certain values that they must take for slots like 'family', i.e. they all have the same family.

- **Logical constraints**

Three types of logical constraints can be applied to slots: symmetricalness, reflexivity and transitivity. Each of these logical constraints limits the values that can be taken as default or 'must have' values for classes, and as values for individuals. Note that these constraints apply to slots that are behaving as *relations* by taking values that are other individuals, effectively linking the individuals together.

- **Symmetricalness**

- Symmetric slots define that if X takes Y as a value, then Y must take X as a value. For example, the slot 'siblings' is symmetric because if Alice is a sibling of Bill then Bill must be a sibling of Alice.
- Asymmetric slots define that if X takes Y as a value, then Y must not take X as a value. For example, the slot 'children' is asymmetric because if Alice is a child of Bill then Bill cannot be a child of Alice.

- **Reflexivity**
 - Reflexive slots define that X always takes itself as a value. For example, the slot 'family' is reflexive because people are always a member of their own family.
 - Irreflexive slots define that X never takes itself as a value. For example, the slot 'parents' is irreflexive because people can never be their own parents.
- **Transitivity**
 - Transitive slots define that if X takes Y as a value and Y takes Z as a value, then X must also take Z as a value. For example, the slot 'ancestor' is transitive because if Bill is an ancestor of Alice's and Clare is an ancestor of Bill's, then Clare must be one of Alice's ancestors.
 - Weakly transitive slots define that if X takes Y as a value and Y takes Z as a value, then X must also take Z as a value, as long as X is not the same as Z. For example, the slot 'siblings' is weakly transitive because if Bill is one of Alice's siblings and Clare is one of Bill's siblings, then Clare must also be one of Alice's siblings. However, Alice is also one of Bill's siblings (by definition), but Alice cannot be said to be one of her own siblings.

Consistency is enforced by not allowing users to assign values that violate the constraints defined on the slot. If inconsistent values have been taken before the constraints are set, this inconsistency is indicated to the user and they are prompted to make changes to either the values or the constraints themselves in order to attain consistency.

- **Inverses**

The inverse of a slot is another slot that encodes the reverse relationship. For example, the inverse of 'children' is 'parents' and the inverse of 'ancestors' is 'descendants' (see Section 6.2.7 for another example). As with 'must have' values, consistency is not enforced on this, but the values that should be taken are indicated to users when they are assigning values to individuals.

Where there is inconsistency, users are prompted to add the values that should be taken to an individual, remove or change the inverse given for the slot, or change the value given for the inverse slot that is causing the values to be inconsistent. For example, if the individual 'Alice' has takes 'Bill' as a value for the slot 'children', 'Bill' and must take the value 'Alice' for the inverse slot 'parents'. If this value is not taken, the user can choose to either change the inverse of the slot 'children' or change the value taken by 'Alice' for the slot 'children'.

- **'Same' slots**

'Same' slots are defined on classes for a slot and indicate other slots that take values that must also be taken by the slot. For example, the slot 'family' must include the values given for the slot 'children', 'parents' and 'siblings', because people's children are part of their family, as are their parents and siblings. As with 'must have' values and values determined by inverses, consistency is not enforced, but values that should be taken are indicated to the user when they assign values.

Where there is inconsistency, users are prompted to add the values that should be taken to an individual, remove or change the 'same' slots that are defined, or change the values given for the slots that are causing the values to be inconsistent. For example, if the individual 'Alice' takes 'Bill' as a value for the slot 'children', 'Alice' must also take the value 'Bill' for the slot 'family'. If this value is not taken, the user can choose to either change the definition giving 'children' as a 'same' slot for 'family', or change the value taken by 'Alice' for the slot 'children'.

- **Cardinality**

The cardinality of a slot determines how many values can be taken for the slot. For example, the minimum cardinality of the slot 'parents' is 2, as all people must have at least two parents (people can have more than two parents if they are adopted or their parents divorce, so the maximum cardinality goes undefined). The minimum and maximum cardinality of the slot 'age', on the other hand, is 1, as all people have an age, and can only have one age (at a time).

The cardinality of a slot can be defined for classes or individuals and is inherited by subclasses and instances. For example, while the cardinality of the slot 'siblings' would generally be undefined, the class 'only children' would have a minimum and maximum cardinality of zero as only children cannot have siblings.

Consistency with inherited cardinality is enforced. The minimum cardinality for a class or individual has to be equal to or greater than the inherited minimum cardinality. Similarly, the maximum cardinality for a class or individual has to be equal to or less than the inherited maximum cardinality. For example, the minimum cardinality of the slot 'siblings' for the class 'people' must be zero in order to allow 'only children' (a subclass of 'people') to have a cardinality of zero.

		Terminology	
		Same	Different
Attributes	Same	Consensus Experts use terminology and concepts in the same way	Correspondence Experts use different terminology for the same concepts
	Different	Conflict Experts use same terminology for different concepts	Contrast Experts differ in terminology and concepts

Table 5.1: Consensus, conflict, correspondence and contrast among experts. From Shaw & Gaines (1989).

Consistency with slot cardinality when assigning values is partially enforced, to the extent that is possible with HTML forms. If the maximum cardinality is 1, users can only select one value; if the minimum cardinality is 1, they are forced to enter a value. Apart from this, users are only reminded of how many values should be taken for the slot.

Where cardinality constraints are violated, users are prompted to meet them or to alter the causes of the cardinality constraints, i.e. the cardinality for the slot for the classes that are defining the cardinality.

If no values are given for an individual or as a default value for a class, these values are inherited from the individual's types or class' superclasses. When slot values are listed or edited, users can click on an icon next to the value to be given an explanation of all the factors that are limiting the values that can or should be taken. From this page, they are able to change any of the aspects affecting the value of the slot (see Section 6.3.10 for an example).

In the explanations given here, I have ignored the possibility of multiple inheritance and the added complications that are involved in the inheritance of possibly different constraints from different superclasses or types. When constraints clash due to multiple inheritance within APECKS, these clashes are made clear to the user and the usual prompts enable them to change the constraints.

5.4.5 COMPARISON SUPPORT

Once roles have been at least partially constructed, the process of comparing them to locate differences and similarities between experts can begin. The comparisons between roles are made using the consensus/conflict/correspondence/contrast classification expounded by Shaw & Gaines (1989) and discussed in Section 2.5.2.1. Table 5.1 shows the four classifications of relationships between conceptual structures as a reminder.

The above classification was intended for use on repertory grids (see Section 2.5.1.3), which have a flat structure where each element has a rating for each construct. The relationships within

ontologies are more complicated than the relationships between grids. Firstly, some of the slots defined within the ontology may not hold values that are suitable for comparisons: these are 'key' values that define an individual uniquely, such as its name or a description of the individual. Secondly, and more importantly, elements within ontologies are not only defined by the values they have in their slots, but also by their position in the class heterarchy. These difficulties also occur in the translation of ontologies into grids for knowledge acquisition, as described in Section 5.4.4.2.

The class structure represented within different roles can be compared in two ways:

1. Recoding the class structure in terms of categorical slots on the individuals represented within the role or boolean rating scales, which represent whether or not an individual is a member of a class.
2. Comparing the classes defined within the ontologies in terms of their instances. In the same way as the names of slots (terminology) and the values that they hold on individuals (attributes) can indicate consensus, conflict, correspondence or contrast between experts, so can the names of classes (terminology) and their instances (attributes). If, within two roles, there are two classes that are named the same and have the same instances as members, this indicates consensus between the experts. If, within two roles, there are two classes that are not named the same, and yet have the same set of instances as members, this indicates correspondence between the experts. Similarly, if there are two classes that are named the same and have different sets of instances as members, this indicates conflict. Having no overlap in either terminology or the set of instances indicates contrast.

APECKS supports both these methods of comparing class hierarchies. It also supports the comparison of slots and slot values by converting roles to grids and using WebGrid-II in a similar manner to that described in Section 5.4.4.2.

The result of these comparisons is a number of pages that indicate how roles compare to one another and prompt users to act to explore any differences between them (see Section 6.4 for examples). At present, until WebGrid or another similar system gives programmatic analysis of its comparisons, users are only given prompts concerning the class hierarchies in the compared roles. They may then change either the name or instances of a class in a role that they own, or write an annotation justifying the hierarchy they have chosen to use.

Figure 5.11 shows part of a page that details the differences in the class structure between two roles. Figure 5.12 shows the WebGrid-II comparison of two roles. Both these figures are from a comparison of two roles: one based on the Gold Standard constructed for the Sisyphus III experiment (the 'Gold Standard' role) and the other on the example on the same material used in Gaines & Shaw (1996a) (the 'WebGrid Example' role).

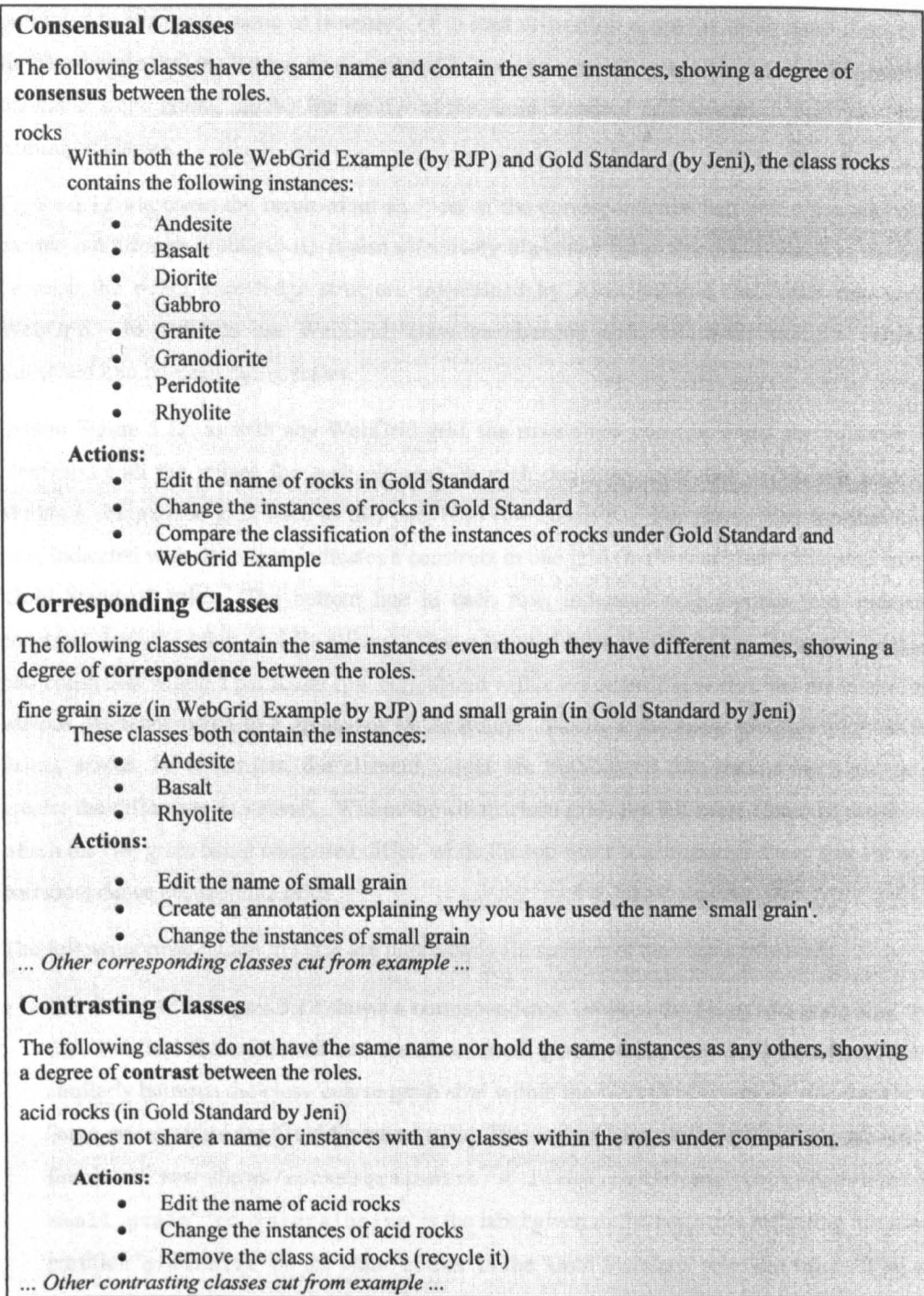


Figure 5.11: Extract from an internal comparison of two roles created using APECKS.

As can be seen in Figure 5.11, users are presented with a classification of the degree of similarity between the classes within the roles. The classification not shown here, 'conflict' occurs when classes in the compared roles share the same name but have different instances. When this occurs, users are shown a table displaying those instances which are in the class within each role individually and which are within both. Wherever classes fall within the classification, users are

prompted to change its name or instances, or to start discussion about the differences if necessary. In the example above, clicking on the prompt "Create an annotation explaining why you have used the name 'small grain'." allows the creator of the 'Gold Standard' role to start discussion about the naming of classes.

Figure 5.12 illustrates the result of an analysis of the correspondence between the same roles as carried out through WebGrid-II. It also effectively illustrates the problems involved in translating between the richer knowledge structure maintained by APECKS and the flatter one used by WebGrid. In order to use WebGrid, class membership and slots with multiple values are translated into boolean rating scales.

Within Figure 5.12, as with any WebGrid grid, the rows show constructs and the columns show elements, with the values for each element on each construct indicated within the grid itself. Within a comparison grid, such as this one, each row consists of two lines. The top line in each row, indicated with italic text, indicates a construct in one grid (in this case that generated from the 'Gold Standard' role). The bottom line in each row, indicated with normal text, indicates a construct from the other grid (in this case that generated from the 'WebGrid Example' role). The two constructs within a particular row correspond with each other: the values that are taken for the various elements match to a greater or lesser extent. Where a particular element takes different values across the constructs, the element values are highlighted (the darker the highlight, the greater the difference in values). Within the comparison grid, the left-most elements are those on which the two grids being compared differ, while the top-most constructs are those that show most correspondence between the grids.

The following rows within the grid are particularly illustrative of the issues involved:

- The first row of Figure 5.12 shows a correspondence between the class 'fine grain size' within the 'WebGrid Example' role and the class 'small grain' within the 'Gold Standard' role, and similarly between the class 'coarse grain size' within the 'WebGrid Example' role and the class 'large grain' within the 'Gold Standard' role. This is read from the labels of the grid. The first line in the row shows 'rocks:grainsize = large grain' and 'rocks:grainsize = small grain'. 'rocks:grainsize' is the label given to the construct reflecting the subclass partition 'grainsize' on the class 'rocks' in the 'Gold Standard' role: the values it takes for these elements are 'small grain' ('sm') or 'large grain' ('la'). The second line shows 'rocks:grain size = coarse grain size' and 'rocks:grain size = fine grain size', which indicate the classes 'coarse grain size' ('co') and 'fine grain size' ('fi') within the subclass partition 'grain size' on the class 'rocks'. Looking at the values within the row, it can be seen that wherever a 'la' appears on the first line, a 'co' appears on the second, and that wherever a 'sm' appears on the first line, a 'fi' appears on the second:

there are no shaded cells in this row. This perfect correspondence is also identified by APECKS, as shown in the page extract above (Figure 2).

- The eighth row of Figure 5.12 shows how different ways of representing the same information can be seen as corresponding to each other. Within the 'WebGrid Example' role, silica content is coded on a rating scale from 1 (high silica content) to 9 (low silica content) while within the 'Gold Standard' role, the corresponding information is represented as four exclusive subclasses in the 'silica content' partition of the class 'rocks': 'acid' ('ac'), 'intermediate' ('in'), 'basic' ('ba') and 'ultra basic' ('ul'). The correspondence between these two representations is not perfect, as indicated by the shaded second column.
- The ninth and thirteenth rows illustrate how users might choose different ways of encoding slots. Within the 'WebGrid Example' role, the colouring of rocks is coded on a rating scale from 1 ('light - leucocratic') to 9 ('dark - melanocratic') whereas within the 'Gold Standard' role, the corresponding slot is categorical, with the three possible values 'melanocratic (dark)', 'mesocratic (medium)' and 'leucocratic (light)'. Within the 'Gold Standard' role, however, the slot 'rock colour' has been given a cardinality of two, so that those rocks that can be medium-or-dark or medium-or-light can be indicated. This is translated into three boolean constructs for WebGrid: 'rock colour:melanocratic (dark)', 'rock colour:mesocratic (medium)' and 'rock colour:leucocratic (light)'. Taking the value '1' for a particular construct indicates that the rock does not take that value for the slot 'rock colour', while taking the value '2' indicates that it does. Each of these three constructs appears within a different row: the fact that the construct representing the taking of the value 'melanocratic (dark)' appears higher than the others indicates that there is most correspondence in the classification of darker rocks within the two roles.
- The tenth row illustrates how comparing roles in this way can indicate causal knowledge that may otherwise be missing. The presence of alkali feldspar, encoded within the 'Gold Standard' role by 'alkali feldspar' being one of the values of the slot 'minerals present', is related to the colour of the rock, encoded within the 'WebGrid Example' role through the 'shade' slot.

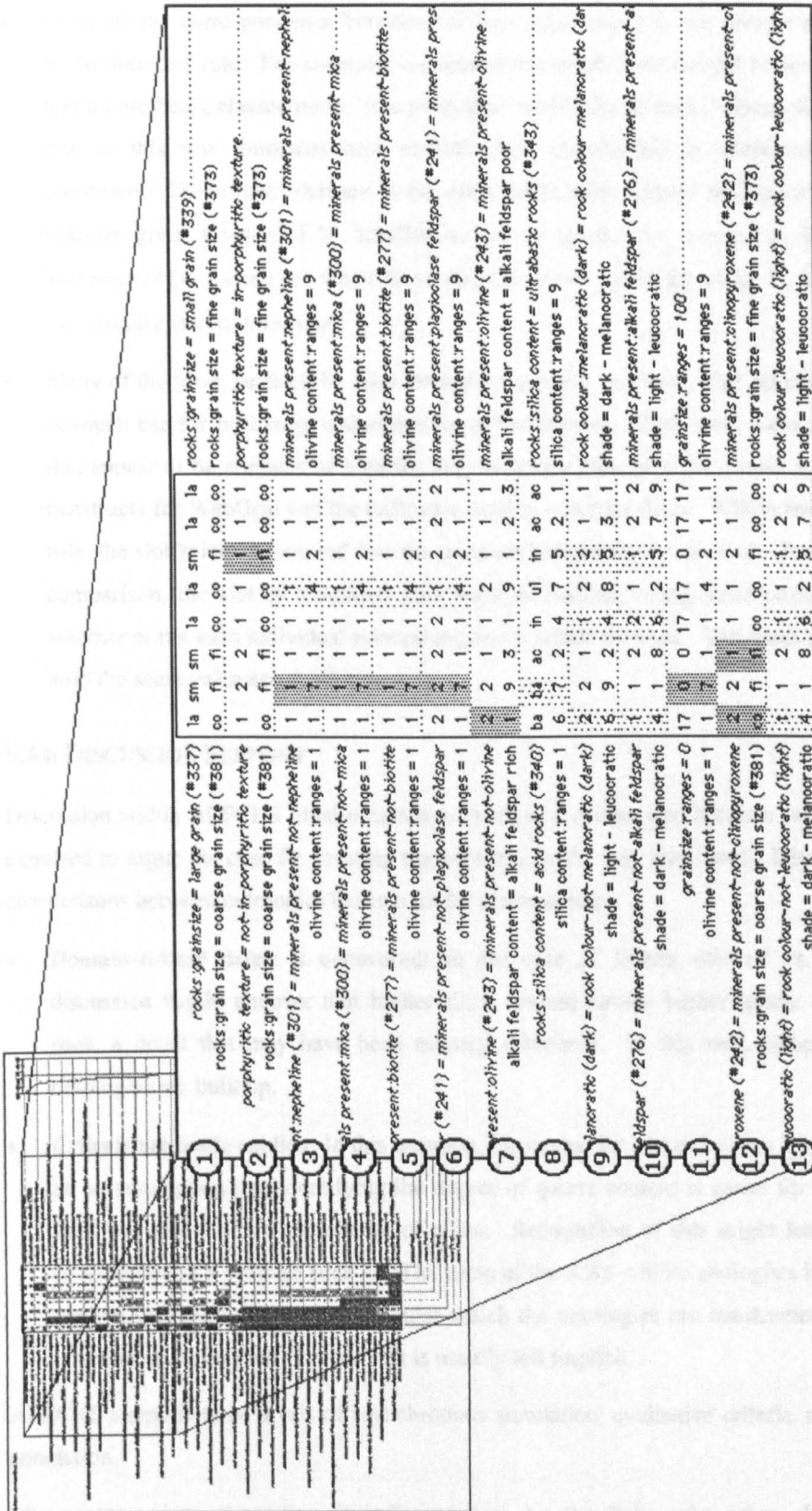


Figure 5.12: WebGrid-II comparison of two roles generated through APECKS.

- Some of the correspondence between the two roles indicates the greater detail within the 'Gold Standard' role. For example, the second row might indicate that rocks with porphyritic texture are being classed under 'fine grain size' within the 'WebGrid Example' role. The first line of this row illustrates how normal class membership is translated into WebGrid constructs. Those rocks that are in the class 'porphyritic texture' within the 'Gold Standard' role are given a value of '2', labelled as 'in-porphyritic texture', for the construct 'porphyritic texture', while those that are not are given the value '1', labelled as 'not-in-porphyritic texture'.
- Many of the rows, such as the third through to seventh, however, give information that is not of much use for furthering understanding of the domain. These rows show correspondences that appear to be artefacts of both the way slots that take multiple values are translated into constructs for WebGrid and the technique used to compare them. Within the 'Gold Standard' role, the slot 'minerals present' lists the minerals present within the rock. For the purposes of comparison, the slot is translated into multiple boolean rating scale constructs indicating whether or not each individual mineral is present within the rock. This leads to constructs that hold the same value across all elements.

5.4.6 DISCUSSION SUPPORT

Discussion within APECKS often occurs as a result of a comparison between two roles: users are expected to argue the case for creating the ontology in the way they have. Discussion based on comparisons between ontologies has two useful consequences:

- **Domain-related detail is uncovered:** In the case of 'quartz content' vs. 'silica content', discussion would uncover that higher silica content causes higher quartz content within a rock, a detail that may have been missing otherwise. In this way, richer, more detailed ontologies are built up.
- **Criteria are made explicit:** In this example, the reason for using the term 'quartz content' may be an assumption that identifying the degree of quartz content is easier for geologists in the field than estimating a percentage of silica. Recognition of this might lead to the explicit construction of a premise such as "The users of the KBS will be geologists in the field." The explicit statement of the criteria under which the ontologies are constructed provides meta-information about their purpose that is usually left implicit.

APECKS supports three levels of asynchronous annotation: evaluative criteria, rationale and free annotation.

The content of an annotation is indicated both by the links with other objects within the knowledge representation and by keywords that can be specified by the user. The objects within

APECKS can include objects in other roles (allowing users to discuss the differences between roles), but not those in other domains. Links between annotations define discussion threads, and are displayed appropriately to indicate this. The keywords that have been used on annotations within a domain are stored so that others commenting in the same domain can select from them as well as define their own keywords. This reduces problems with people simply phrasing the same conceptual keyword in a different way.

All annotations can be searched in terms of when they were made, who made them, and the keywords and objects associated with them (see Section 6.4.5 for an example). The context in which such a search is made affects the search, so that the results of searching for changes made by Jeni in one role would be different to that made in another role. The most general context in which annotations can be searched for is the entire domain.

5.4.6.1 Evaluative Criteria

As discussed in Section 5.3.3.4, using APECKS can be thought of as conducting a Design Space Analysis. Within APECKS, the overall design question is 'How should the domain be represented?' and each role created by an expert can be seen as an option in answer to that question. Sub-questions relate to the structure of the ontologies that are created, such as 'Which individuals belong to this class?', 'What are the slots on this class?' and 'What is the value for this slot on this individual?'. The questions and the options that are created are thus implicit within APECKS. The criteria used to create the options are also implicit to each user unless they choose to make them explicit in order to justify the way they have structured one of their roles. APECKS allows the users to generate a set of criteria, each of which can operate across any number of roles (see Section 6.5.2 and Section 6.5.7 for examples). Each role likewise can define explicitly a number of criteria under which it was created.

5.4.6.2 Rationale

Every change made to an object is recorded and can be accessed by anyone (see Section 6.2.8 for an example). Such a record grows extremely large, both to store and to browse. Changes that undo previously recorded changes 'cancel out' the previous change. In addition, a limit on the number of changes that are recorded can be set. To facilitate the browsing of changes, there is a context-sensitive search facility just as there is for annotations and criteria (see Section 6.5.5 for an example).

Each change made to an object can be freely annotated by anyone. This facility can be used to explain the reasons they made a particular change or to question someone else's change. In this way, a descriptive rationale of the changes made over time can be built up. Sections 6.2.8 and 6.5.4 give examples of this.

5.4.6.3 Free Annotation

Formal communication often limits the expressiveness of discussion and can be subverted by users to enable them to make their point, which detracts from its utility. Informal or semi-formal annotation enables the users of the system to explore their reasoning more fully and to bring together items that appear to be disparate to the system but that can be seen to be similar to human intelligence. This full discussion might then lead on to the construction of evaluative criteria. Sections 6.5.2 and 6.5.7 gives examples of this.

5.5 CONCLUSIONS

Table 5.2 lists the requirements specified in Section 5.3.3, showing how they are fulfilled within APECKS and an illustrative pointer to Chapter 6.

APECKS is designed to be useful to two overlapping but distinct groups of people:

- **Knowledge Engineers**

APECKS offers the same benefits of other ontology browsers to knowledge engineers (albeit in a more limited manner) - that of creating, browsing and storing ontologies in a network-accessible manner. On top of this, knowledge engineers can use APECKS to compare and discuss ontologies that they have constructed and to identify the implicit criteria they use in their construction, leading to methodologies that are more specific.

- **Domain Experts**

APECKS offers a knowledge-oriented way of structuring and recording communication about a domain, with support for knowledge acquisition techniques. This has utility in organisational memory, design rationale, teaching and other tasks involving the recording of opinion.

This chapter has detailed the rationale behind the design of APECKS and described those features intimately involved with the main lifecycle of APECKS, the creation, comparison and discussion of personal ontologies. The next three chapters focus on APECKS in its current use and future development.

- Chapter 6 walks through four scenarios of use of APECKS, to give an idea of how the system works in use and its applicability for a number of tasks.
- Chapter 7 presents the results of an evaluation study in which domain experts used APECKS to create ontologies in the domain 'Mammals'.
- Chapter 8 outlines some of the developments that could take APECKS forward in the future.

Requirement	Provided by	Illustrated in
1. Users should have persistent identities	Use of MOOs: they include persistent user representations	-
2. User information should be stored and accessible to other users	Adaptations to MOO database for storing extra information about users	-
3. Domains should be considered in isolation and information about them stored explicitly	Separate objects representing domains (see Section 5.4.2)	Sections 6.2.2; Section 6.4.2
4. Knowledge should be represented internally in a semi-formal knowledge representation	Internal knowledge representation described in Section 5.4.2.	-
5. The internal knowledge representation should be transformable into a human-readable presentation	Dynamic generation of HTML pages (see Section 5.4.1)	Chapter 6
6. Users should be able to browse through the available knowledge	Links and buttons (see Section 5.4.3.3)	Chapter 6
7. Users should be able to construct and edit knowledge representations	'Edit' pages (see Section 5.4.3.2)	Section 6.2
8. Roles should be considered separately but retain a link to other roles within the same domain	Separate objects representing roles (Section 5.4.2)	Section 6.3.2
9. Users should be led step-by-step through knowledge acquisition process	Internal and external KA support (see Section 5.4.4)	Section 6.3
10. The system should support contrived knowledge acquisition techniques for direct knowledge acquisition from users	Internal and external KA support (see Section 5.4.4)	Section 6.3
11. Knowledge representations should be compared in an automated fashion	Internal and external role comparison (see Section 5.4.5)	Section 6.4
12. Prompts should be generated for users to make changes to and discuss roles on the basis of comparisons	Internal role comparison (see Section 5.4.5)	Section 6.4.3
13. Evaluative criteria should be represented within the system and viewable by users	Separate objects representing criteria (see Section 5.4.6.1)	Section 6.5.2
14. Arguments concerning the validity of criteria and the applicability of criteria to roles should be represented within the system and viewable by users	Separate objects representing annotations (see Section 5.4.6.3)	Section 6.5.7
15. Users should be able to communicate synchronously	Use of MOOs: they include synchronous communication	-
16. Users should be able to indicate areas under discussion directly within communication	Attachment of annotations to objects (see Section 5.4.6.3)	Section 6.2.8
17. Discussion between users should be archived for future use	Permanence of annotations and criteria (see Section 5.4.6)	Section 6.4.5
18. Discussion archives should be searchable	Context-sensitive search forms for annotations and criteria (see Section 5.4.6)	Section 6.4.5
19. Changes to roles and to discussion threads should be recorded	Automatic recording of all changes (see Section 5.4.6.2)	Section 6.5.5
20. Users should be able to construct a rationale explaining the changes made	Provision of explanations for changes and KA prompts not followed (see Section 5.4.4.1 and 5.4.6.2)	Section 6.2.8; Section 6.3.6
21. Users should be able to browse, discuss and search the rationale	Searchable archives of changes (see Section 5.4.6.2)	Section 6.5.5

Table 5.2: The requirements specified in Section 5.3.3, showing how they are fulfilled within APECKS and illustrative pointers to Chapter 6.

CHAPTER 6 SCENARIOS

6.1 SUMMARY

The last chapter described APECKS as a system in technical terms. This chapter describes a number of situations in which APECKS is used in different ways to achieve different goals, each of which is, at heart, collaborative ontology construction. Four scenarios are described in this chapter:

1. Two knowledge engineers use APECKS to create, share and maintain ontologies in the domain of igneous rocks as part of the Sisyphus III experiment. This scenario focuses on support for the construction of ontologies by knowledge engineers.
2. A student uses APECKS to aid in revision, again in the geology domain. The student, with no knowledge engineering experience uses the knowledge elicitation support built into APECKS in order to construct a representation of her knowledge. This scenario focuses on support for the construction of ontologies by domain experts.
3. A researcher uses APECKS to compare mammalian taxonomies since the 18th century. Using APECKS to identify differences between the taxonomies, he documents their differences and makes the resulting information available to his students. This scenario focuses on support for the comparison of ontologies.
4. The knowledge acquisition community use APECKS to create an XML schema for representing information about their community. The participants use APECKS to create alternative schemas and comment on those that others put forward, all the while building a rationale for their final schema. This scenario focuses on support for the discussion of ontologies.

The scenarios display some of APECKS's useful features, what can be achieved using it, and give an impression of how it feels to use the system.

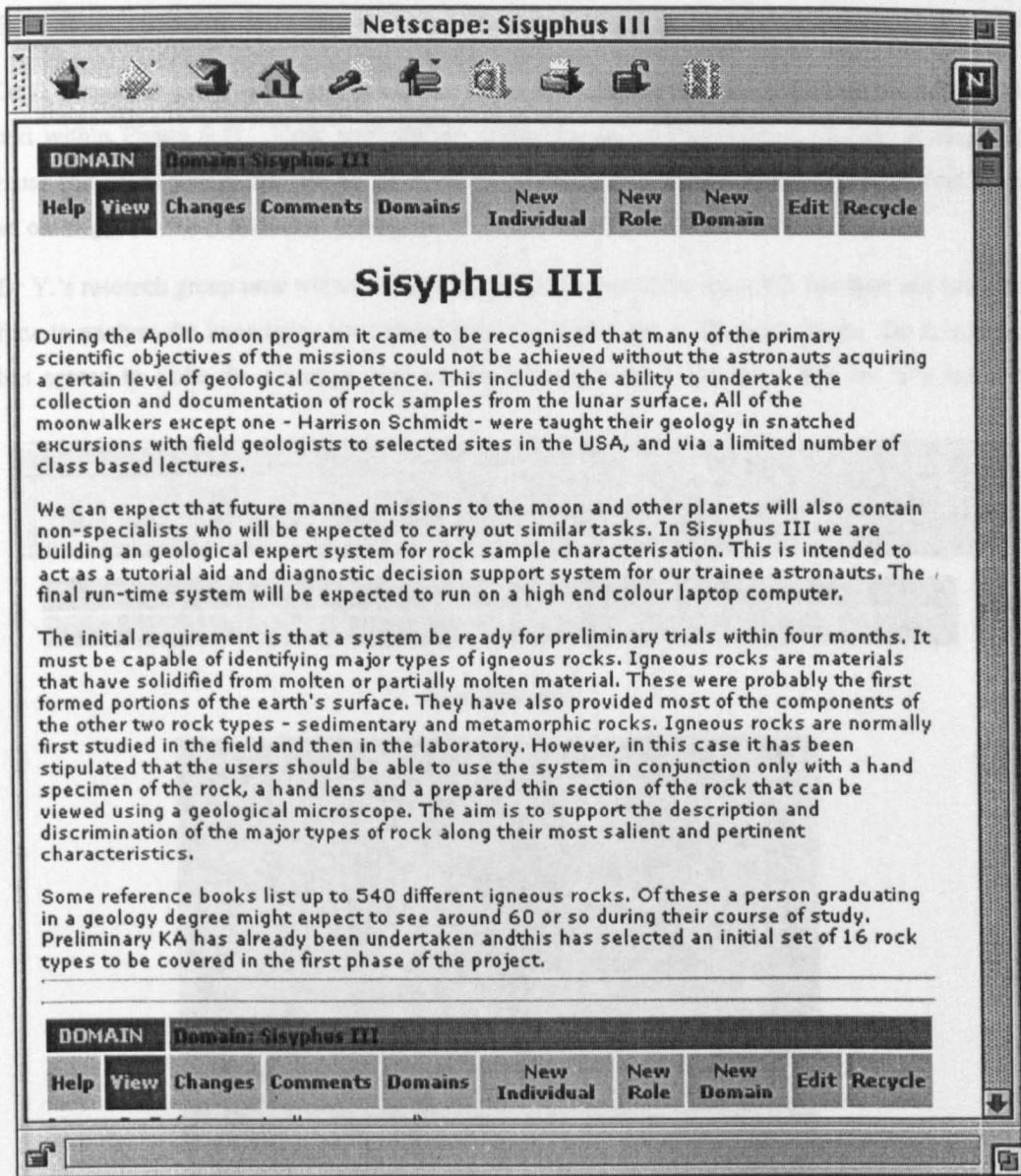


Figure 6.1: The new domain created by Dr Z., including a description of the Sisyphus III experiment.

6.2 CONSTRUCTION: APECKS AS AN ONTOLOGY SERVER

This scenario stresses the use of APECKS by knowledge engineers to construct ontologies. It explores the traditional aspects of APECKS as an ontology server: a means to share ontologies between collaborating knowledge engineers. Since the users in this scenario are knowledge engineers, they are expert enough in knowledge engineering to take advantage of the methods for directly manipulating ontologies. The construction techniques discussed here contrast with the supported knowledge acquisition for use by domain experts demonstrated in Section 6.3. They can also be used to construct roles to be compared as in Section 6.4 or to be discussed as in 6.5.

6.2.1 BACKGROUND

Dr Z.'s research group is taking part in the Sisyphus III experiment (a description is given in the text within Figure 6.1). Their methodology promotes the utility of ontologies for sharing and reuse and so, as part of the first phase of the study, they invested their time in the construction of an ontology on which their first system was based.

Dr Y.'s research group now wishes to take part in phase two of Sisyphus III, but does not have the time to analyse the knowledge acquisition from the first phase of the experiment. Dr Z.'s group has agreed to make the ontology they constructed available, in the hope that Dr. Y's research

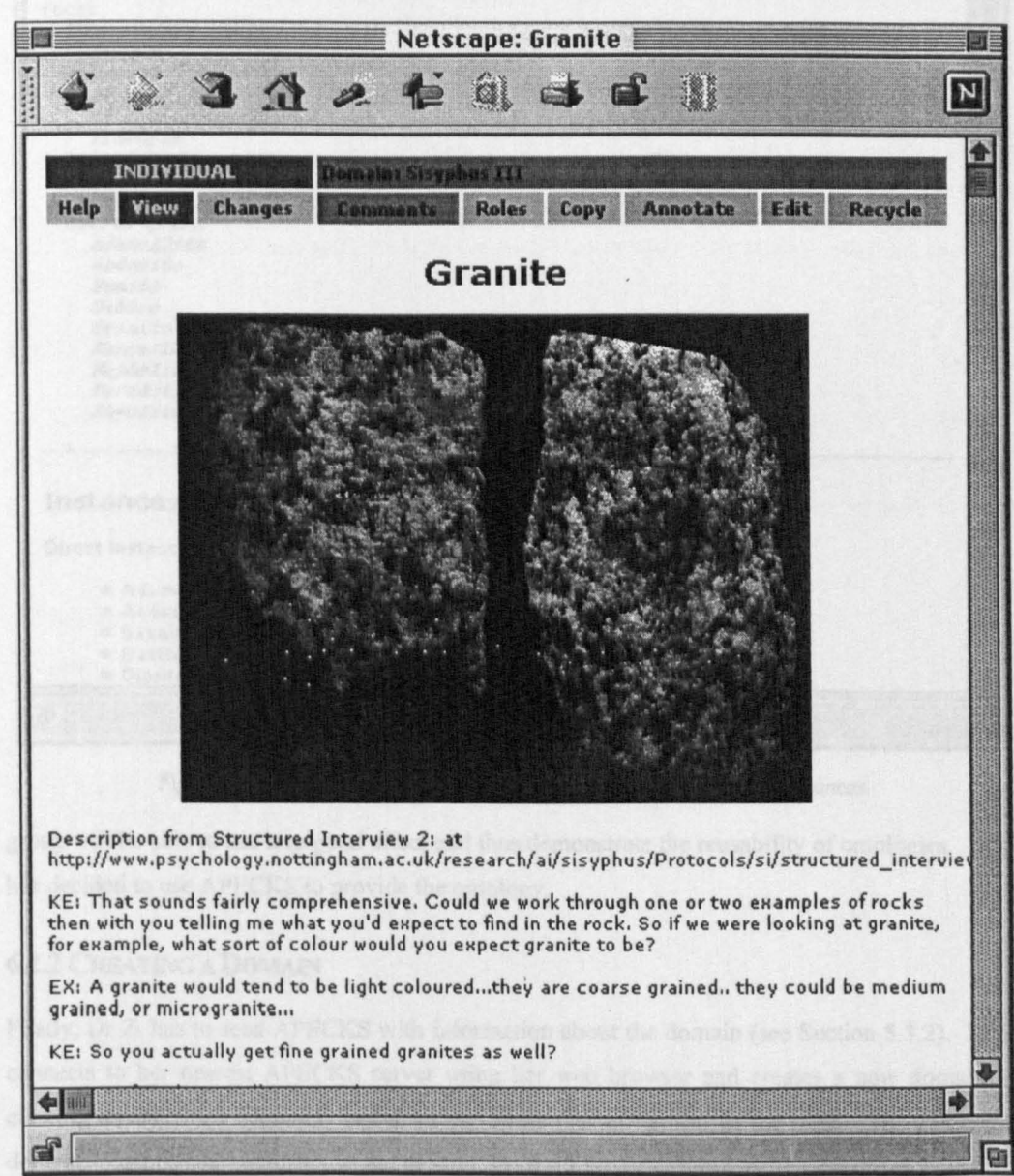


Figure 6.2: The newly created individual, Granite.

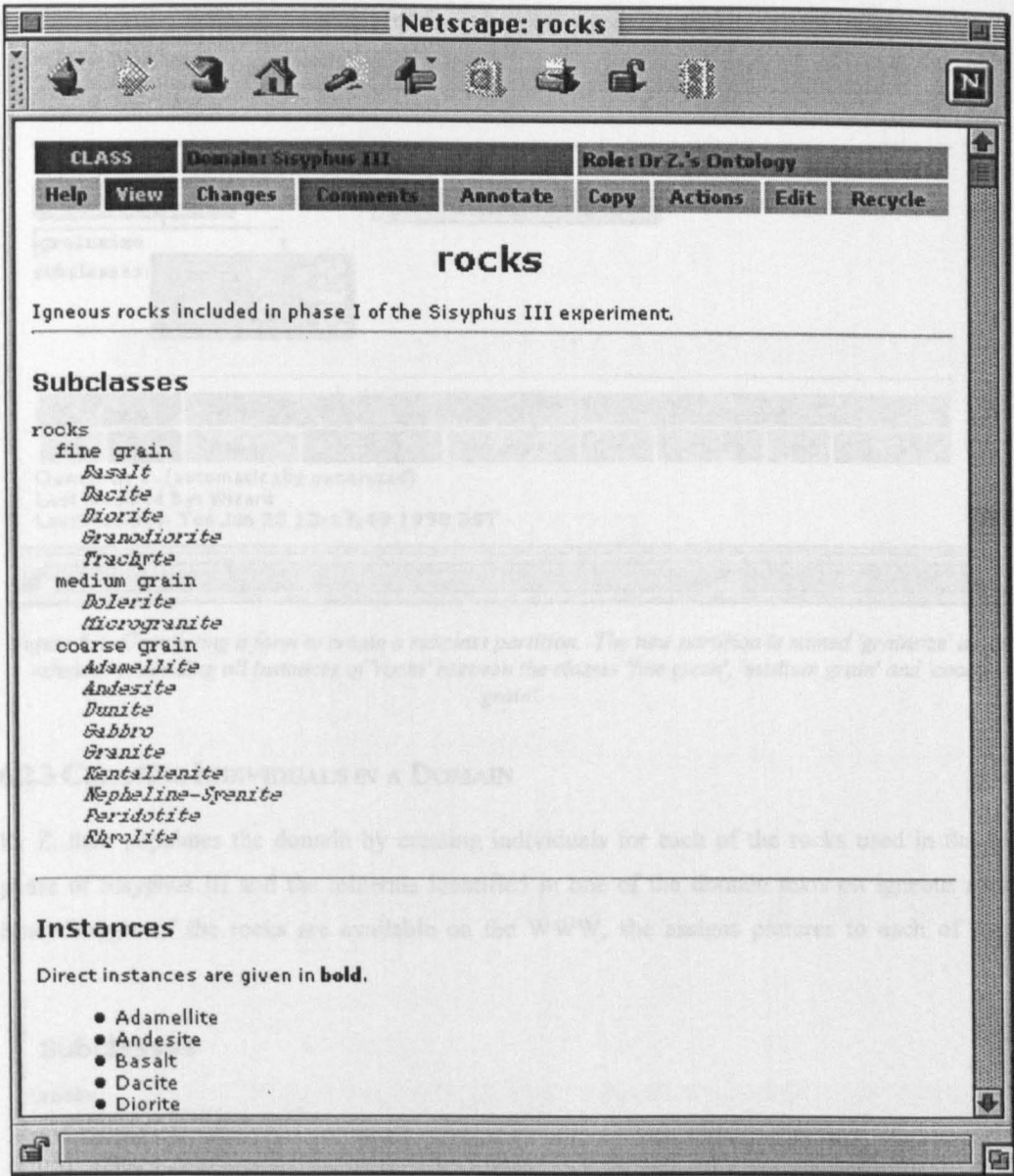


Figure 6.3: The class rocks, showing its subclasses and a list of its instances.

group will be able to use it to good effect and thus demonstrate the reusability of ontologies. Dr Z. has decided to use APECKS to provide the ontology.

6.2.2 CREATING A DOMAIN

Firstly, Dr Z. has to seed APECKS with information about the domain (see Section 5.3.2). Dr Z. connects to her nearest APECKS server using her web browser and creates a new domain by clicking on the ‘New Domain’ button in the button bar at the top of the page. She names the domain ‘Sisyphus III’ and fills in the description from the description of the experiment. Figure 6.1 shows the domain at this stage.

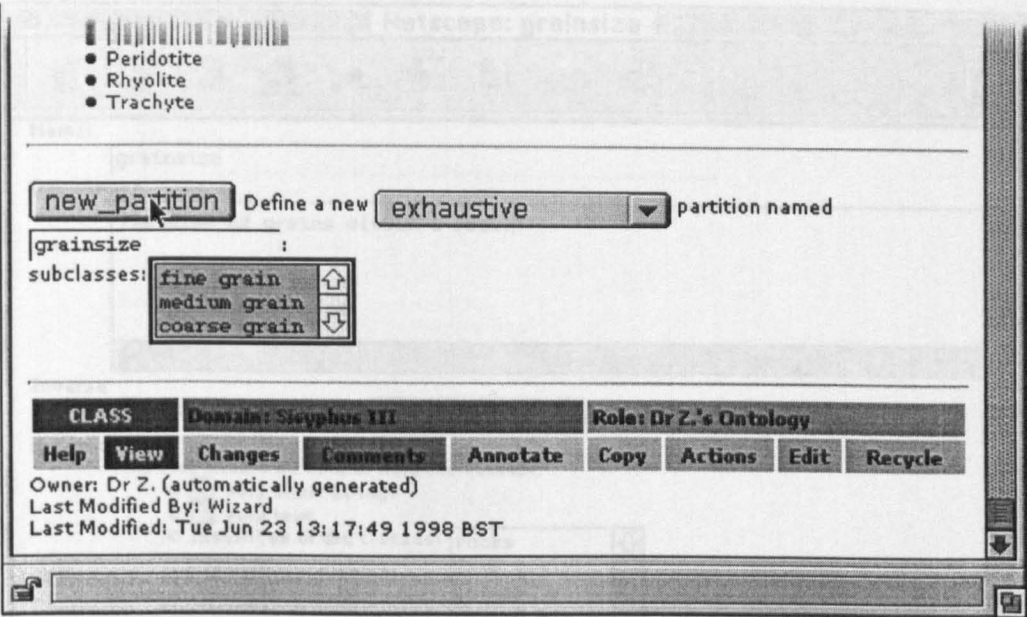


Figure 6.4: Completing a form to create a subclass partition. The new partition is named 'grainsize' and is exhaustive, dividing all instances of 'rocks' between the classes 'fine grain', 'medium grain' and 'coarse grain'.

6.2.3 CREATING INDIVIDUALS IN A DOMAIN

Dr Z. then populates the domain by creating individuals for each of the rocks used in the first phase of Sisyphus III and the minerals identified in one of the domain texts on igneous rocks. Since images of the rocks are available on the WWW, she assigns pictures to each of them.

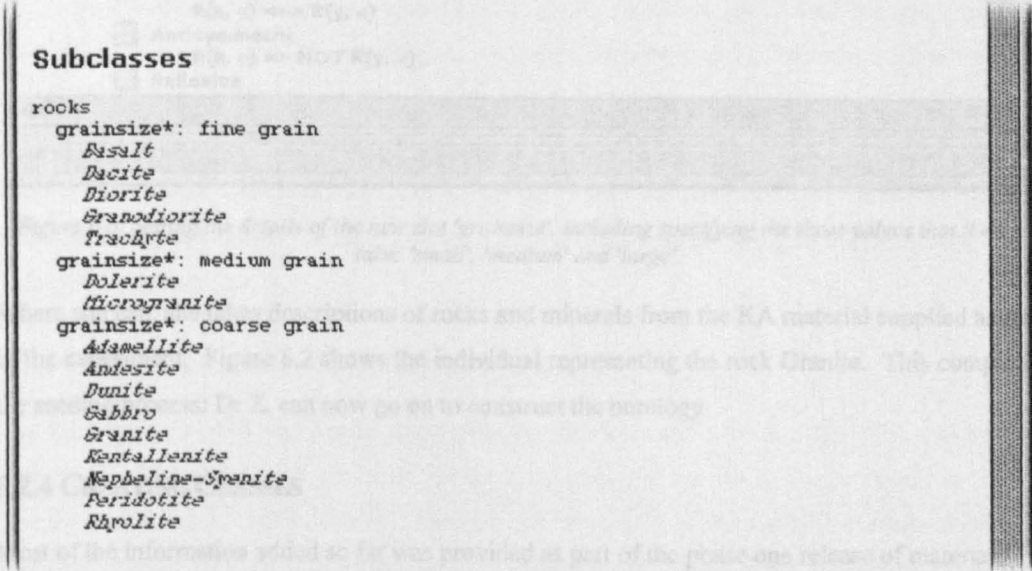


Figure 6.5: The class hierarchy after the subclass partition has been created. The name of the subclass partition appears next to the names of the subclasses in it, with the asterisk indicating that the subclass partition is exhaustive.

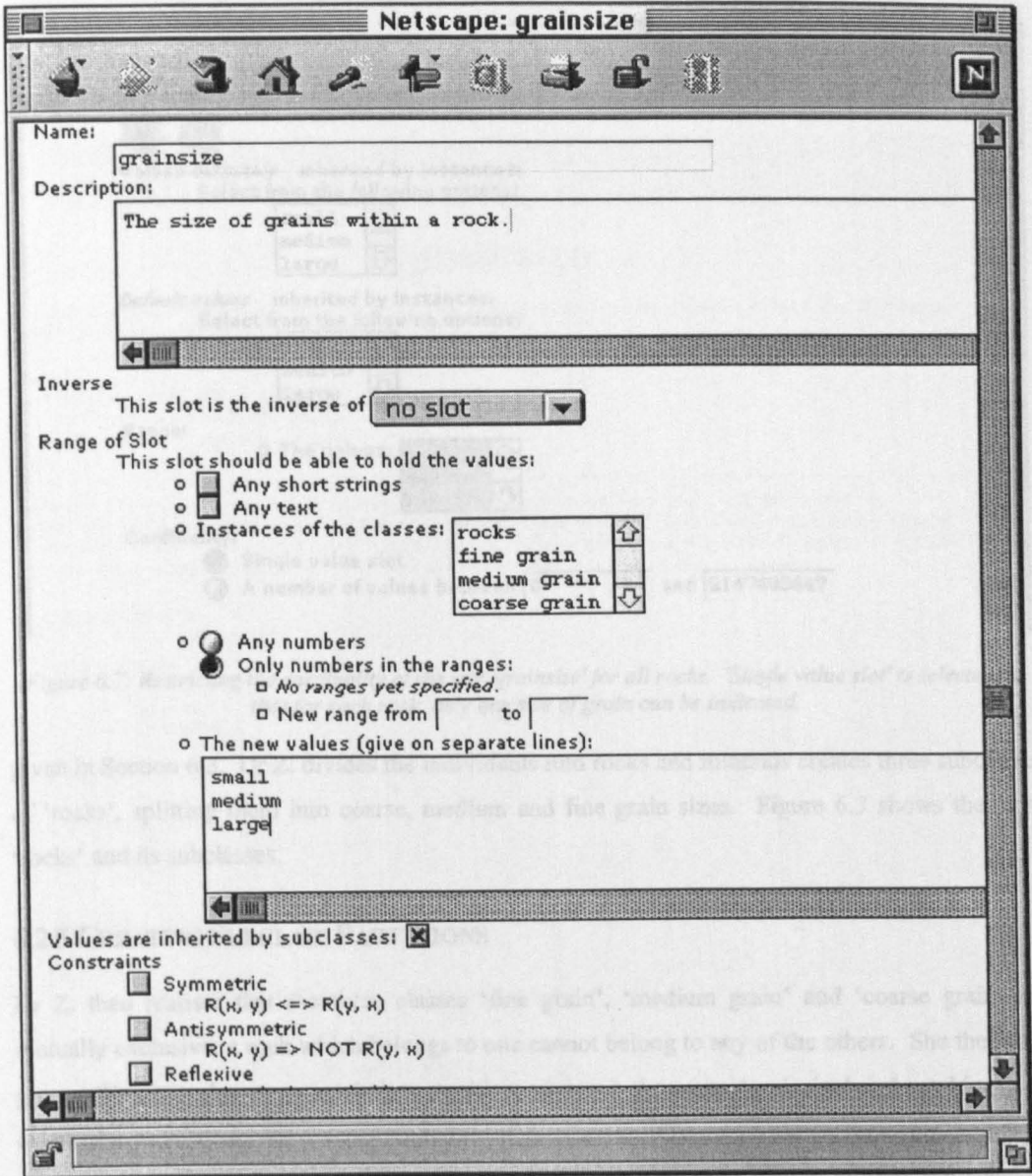


Figure 6.6: Setting the details of the new slot 'grainsize', including specifying the three values that it may take: 'small', 'medium' and 'large'.

Where she can, she takes descriptions of rocks and minerals from the KA material supplied as part of the experiment. Figure 6.2 shows the individual representing the rock Granite. This completes the seeding process: Dr Z. can now go on to construct the ontology.

6.2.4 CREATING CLASSES

Most of the information added so far was provided as part of the phase one release of material. Dr Z. now creates a role for her research group's ontology and starts to build up the ontology that they used within their system. Note that due to Dr Z.'s knowledge engineering expertise, she does not use the knowledge acquisition prompts available within APECKS: examples of their use are

Class

☒ Update rocks

Values *definitely* inherited by instances:
Select from the following options:

small

medium

large

Default values inherited by instances:
Select from the following options:

small

medium

large

Range:

☐ The values:

small

medium

large

Cardinality:

☒ Single value slot

☐ A number of values between

0

 and

2147483647

Figure 6.7: Restricting the cardinality of the slot 'grainsize' for all rocks. 'Single value slot' is selected, so that for each rock, only one size of grain can be indicated.

given in Section 6.3. Dr Z. divides the individuals into rocks and minerals creates three subclasses of 'rocks', splitting them into coarse, medium and fine grain sizes. Figure 6.3 shows the class 'rocks' and its subclasses.

6.2.5 CREATING SUBCLASS PARTITIONS

Dr Z. then realises that the three classes 'fine grain', 'medium grain' and 'coarse grain' are mutually exclusive: a rock which belongs to one cannot belong to any of the others. She therefore groups the three classes as a subclass partition, giving it the name 'grainsize' and marking it as exhaustive as all rocks have to belong to one of the classes. Figure 6.4 shows the completed form used to create a subclass partition and Figure 6.5 shows the class hierarchy after the subclass partition has been created. Other ways of creating subclass partitions using APECKS are shown in Section 6.3.4 and 6.3.11.

6.2.6 CREATING SLOTS WITH DEFAULT VALUES

Dr. Z. decides to build some redundancy into the ontology and creates a slot that also encodes information about the rock's grainsize. She sets the range of the slot to the three values 'small', 'medium' and 'large' (see Figure 6.6) and sets the cardinality of the slot (the number of possible values it can take) to one (see Figure 6.7). Section 5.4.4.3 describes the logical implications of setting ranges and cardinalities. Then, for each of the three classes 'fine grain', 'medium grain' and 'coarse grain', she sets the default value of the slot to be the appropriate grainsize.

130

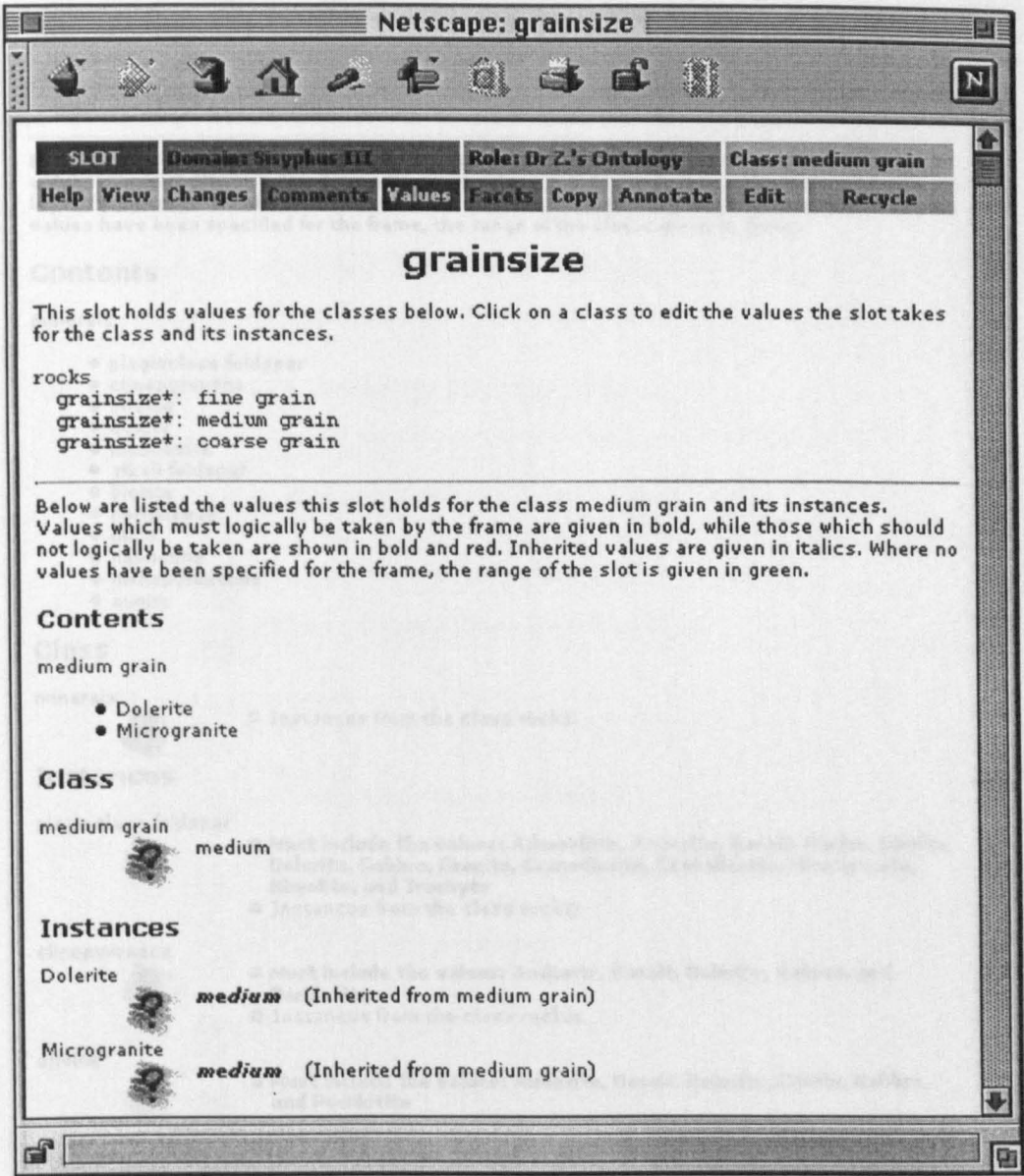


Figure 6.8: A list of values for 'grainsize' for instances of the class 'medium grain'. Note that the rocks listed have inherited the value 'medium' for the slot 'grainsize' from the class 'medium grain'.

and 'coarse grain', she sets relevant default value for grainsize. Figure 6.8 shows how the values for 'grainsize' are inherited by the rocks.

6.2.7 CREATING SLOTS WITH INVERSES

In the process of continuing to create the ontology, Dr Z. creates a slot for all rocks, 'contains', which lists the minerals present within the rock. She sets the range of this slot to be any individual within the class 'minerals'. She goes through the rocks and selects the minerals present in each. It then occurs to her that it is useful also to know which rocks a particular mineral occurs in, so she creates a slot, 'present in', on the class 'minerals'. She sets this slot to be the inverse of 'contains',

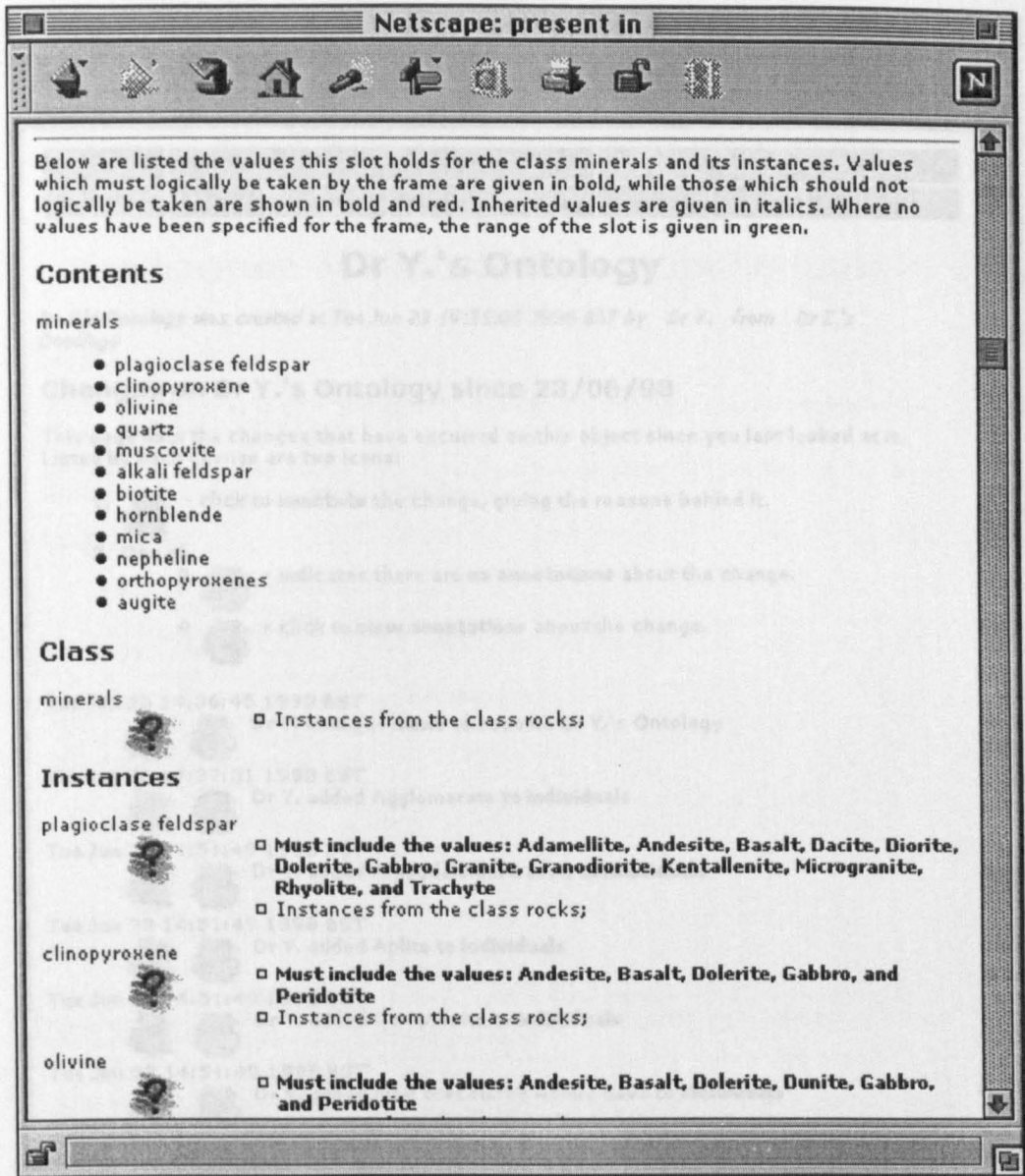


Figure 6.9: The values of the slot 'present in' for minerals after it has been identified as the inverse of the slot 'contains'. Note that although the minerals must logically be present in the rocks listed, they do not actually take the value, which is why the range 'Instances from the class rocks' is indicated for each.

and the minerals automatically list the rocks they are contained within as values that must logically be taken (see Section 5.4.4.3). The values for 'present in' can be seen in Figure 6.9.

6.2.8 CREATING RATIONALES

When she has completed the ontology, Dr Z. lets Dr Y. know the URL of the ontology and wishes him luck (see Section 5.4.3.2). When the second phase material is released, Dr Y. sees that the sixteen new rocks are not included within the ontology with which Dr Z. provided him. He therefore makes a copy of Dr Z.'s ontology (this is done with a single mouse-click: Dr Z. does not

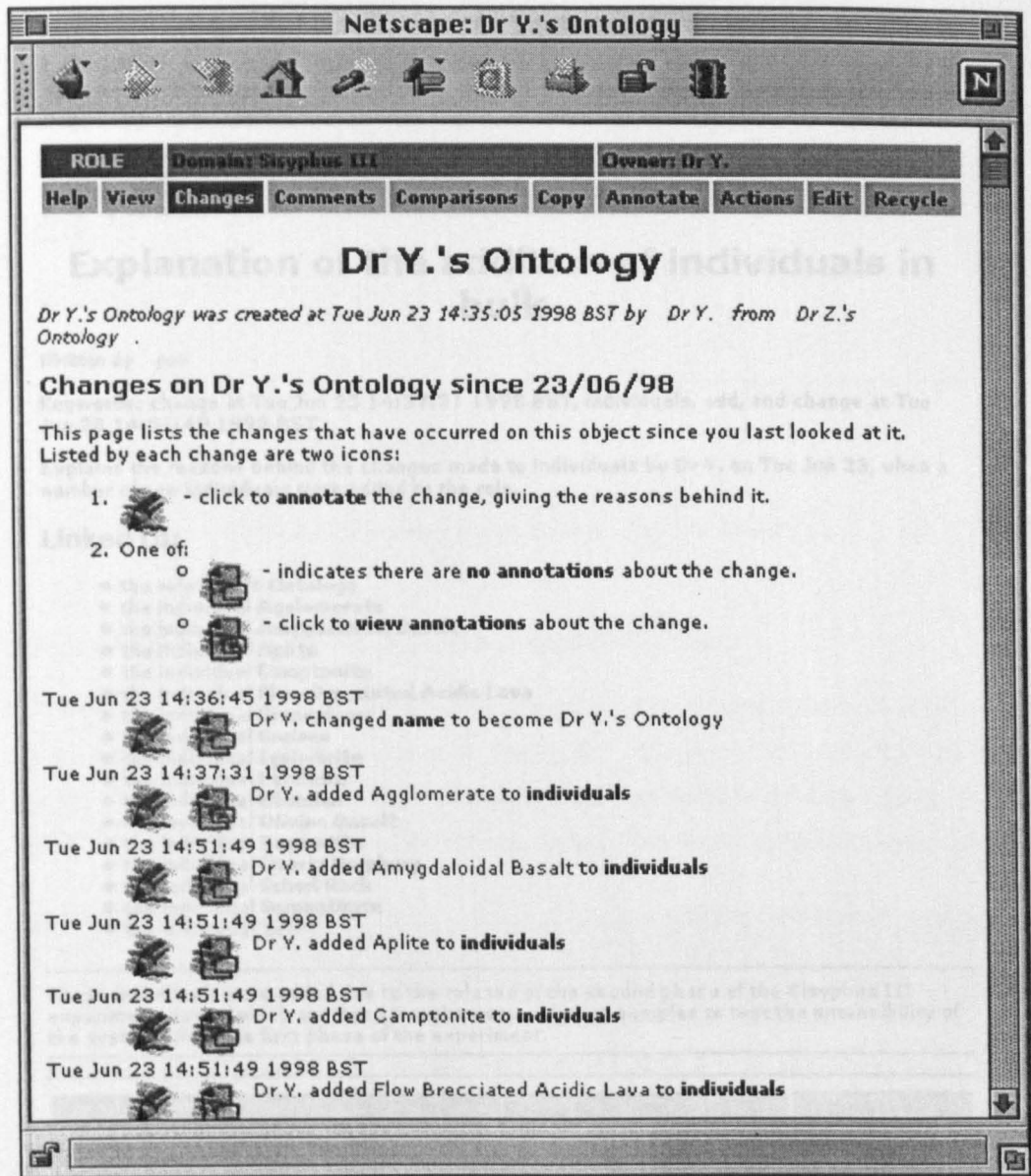


Figure 6.10: The list of changes to Dr Y.'s ontology after copying it originally from Dr Z.'s ontology. Dr Y. has already attached an annotation to the addition of Agglomerate to the role.

have to copy the ontology by hand), creates more individuals representing the new rocks, and adds them to it.

Dr Y. and Dr Z. have agreed to keep track of any changes that need to be made to the ontology, so they can write it up in a paper on ontology reuse later. When he has added the new rocks, he looks at the changes made to the ontology. He creates an annotation on the first change that notes that the new rocks were added as a response to the phase two release of material (see Section 5.4.6.2). He adds the same annotation to the other changes in which rocks were added. The list of changes made in Dr Y.'s ontology can be seen in Figure 6.10, and the annotation written about them in Figure 6.11.

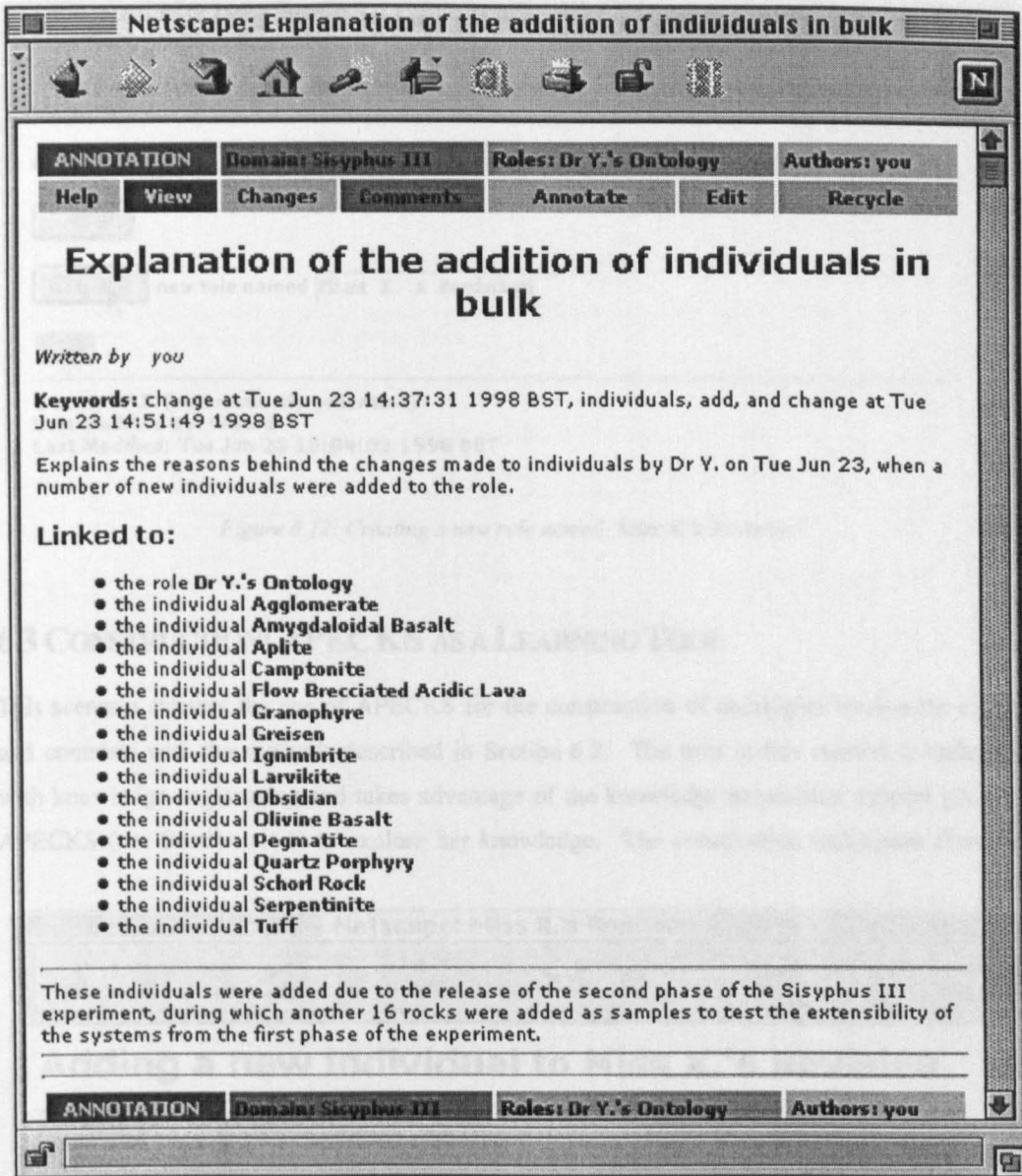


Figure 6.11: The annotation explaining the addition of extra rocks to Dr Y.'s ontology.

Dr Y. continues to make changes to his copy of the ontology, updating it within information from the set of knowledge acquisition materials released in the second phase of Sisyphus III. Dr X. monitors the changes every now and then and the knowledge engineers occasionally discuss the finer points of ontology engineering as demonstrated in their two approaches. Full examples of the type of exchanges that might go on between the knowledge engineers are explored in Section 6.4 and Section 6.5.

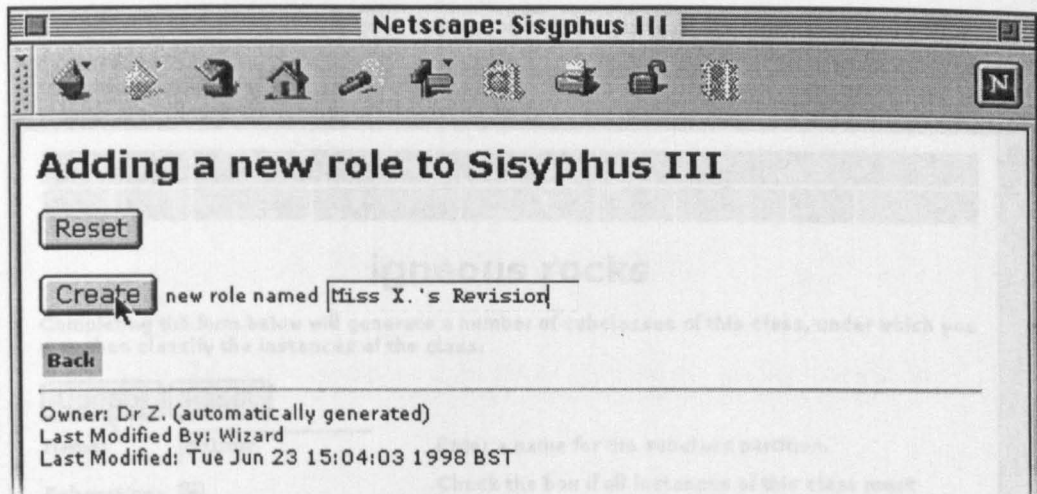


Figure 6.12: Creating a new role named 'Miss X.'s Revision'.

6.3 CONSTRUCTION: APECKS AS A LEARNING TOOL

This scenario stresses the use of APECKS for the construction of ontologies by domain experts and contrasts with the methods described in Section 6.2. The user in this context is unfamiliar with knowledge engineering and takes advantage of the knowledge acquisition support given by APECKS (see Section 5.4.4) to explore her knowledge. The construction techniques discussed

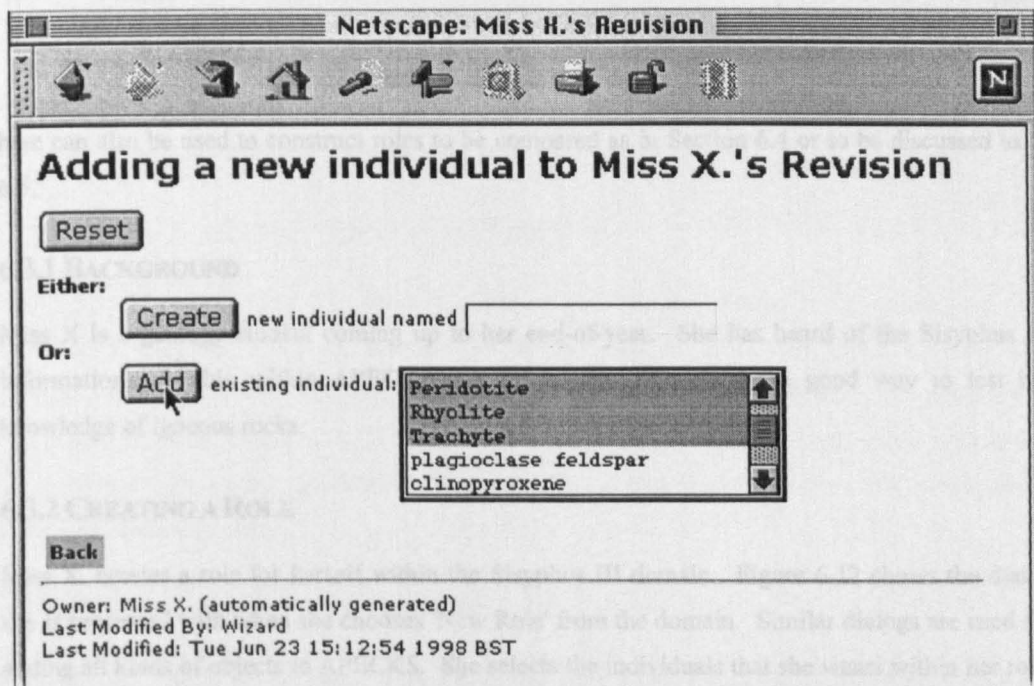


Figure 6.13: Adding individuals to Miss X.'s role. Note that she selects only the rocks and does not worry about the minerals for the moment.

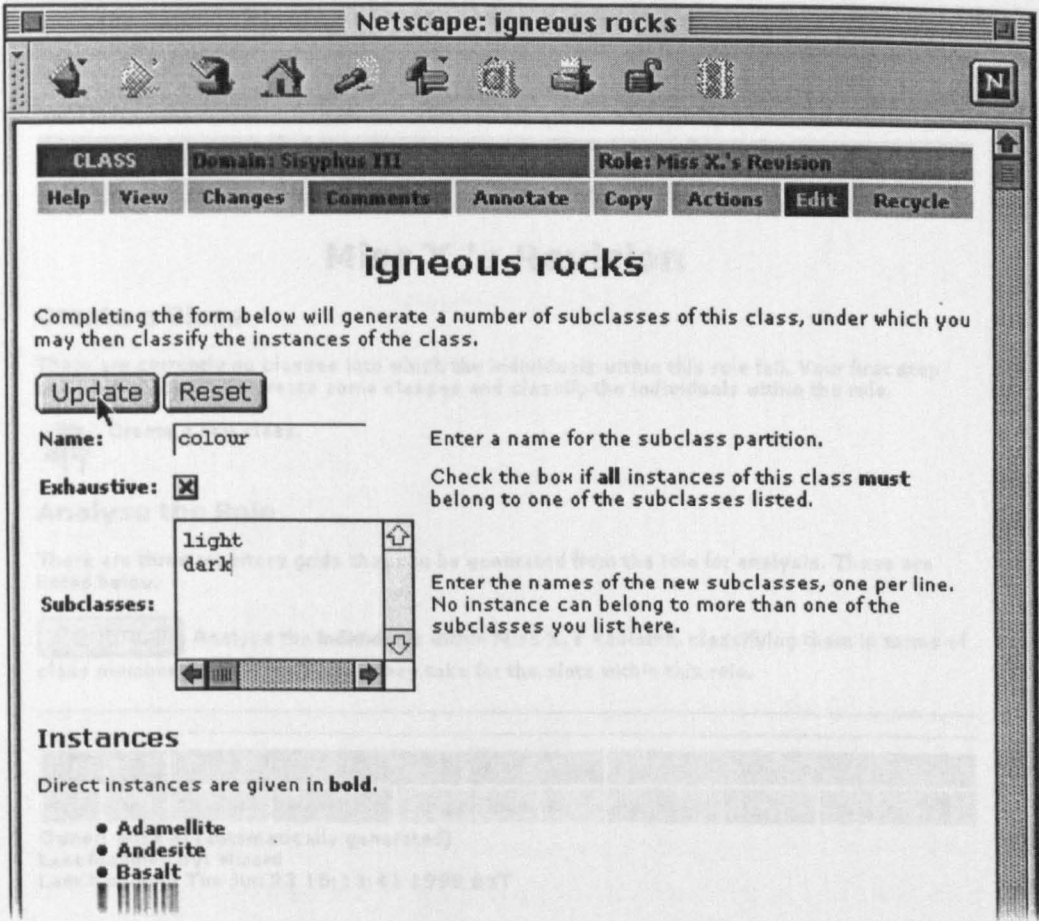


Figure 6.14: A completed form for creating an exhaustive subclass partition named 'colour' with the subclasses 'light' and 'dark'.

here can also be used to construct roles to be compared as in Section 6.4 or to be discussed as in 6.5.

6.3.1 BACKGROUND

Miss X is a geology student coming up to her end-of-year. She has heard of the Sisyphus III information available within APECKS and thinks that it might be a good way to test her knowledge of igneous rocks.

6.3.2 CREATING A ROLE

Miss X. creates a role for herself within the Sisyphus III domain. Figure 6.12 shows the dialog she is presented with when she chooses 'New Role' from the domain. Similar dialogs are used for adding all kinds of objects to APECKS. She selects the individuals that she wants within her role, choosing only the rocks and ignoring the minerals as she does not want to get into them at this stage. Figure 6.13 shows Miss X. selecting individuals for her new role. APECKS allows

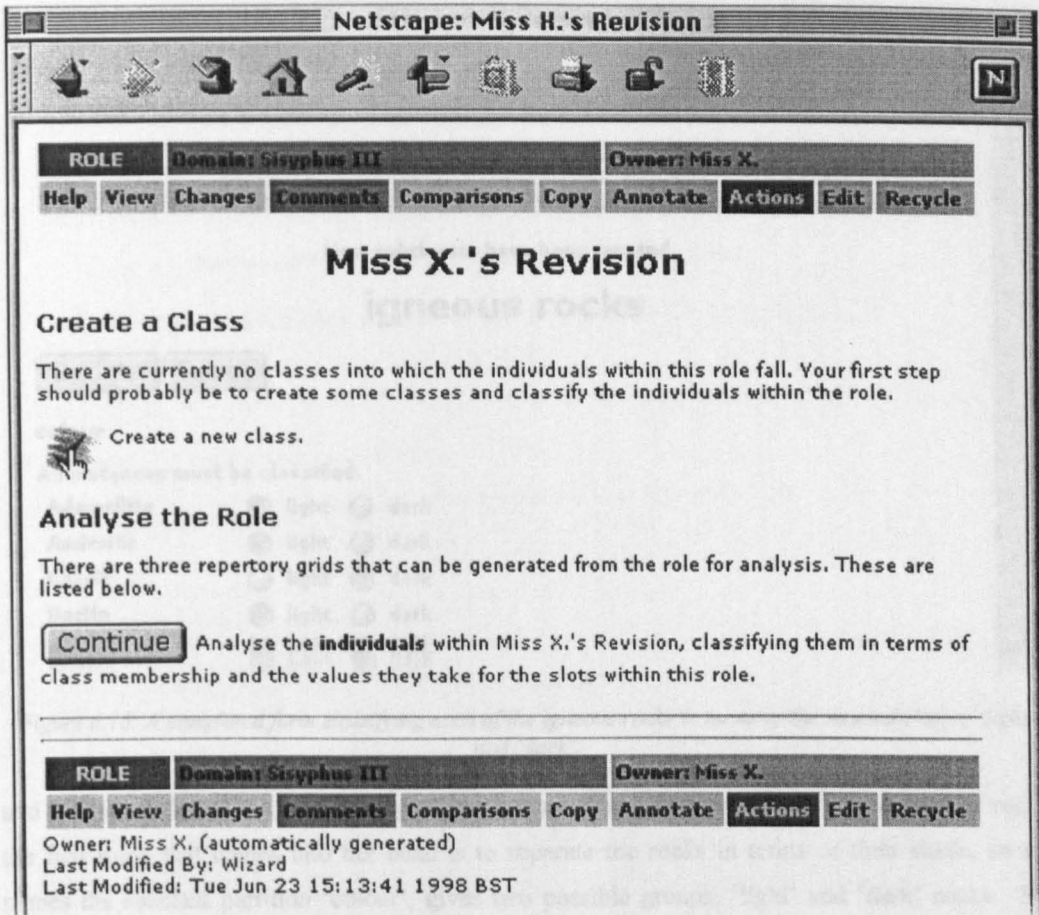


Figure 6.15: The initial set of actions suggested to Miss X.. She opts for creating a new class.

different users to focus on different aspects of a particular domain by selecting only some of the individuals available within it for discussion in their own role.

6.3.3 USING KNOWLEDGE ELICITATION PROMPTS

From the main view of her role, Miss X. clicks on the 'Actions' button in the button bar. The 'Actions' view gives a list of prompts for naïve users to help them build a knowledge representation and keep it internally consistent (see Section 5.4.4.1). Miss X. is prompted to create a class for the individuals. Figure 6.15 shows the prompts Miss X. is presented with at this stage. She creates a class into which all the rocks fit, 'igneous rocks' and puts all the individuals in her role within the class.

6.3.4 CREATING SUBCLASS PARTITIONS

Miss X. then selects the 'Actions' view again, this time of the class, and is told that the class is quite large and asked whether she wants to create some subclasses. Prompts like this expand the ontology by adding new concepts (see Section 5.4.4.1). She decides to do create some subclasses

CLASS **Domain: Sisyphus III** **Role: Miss X.'s Revision**

Help **View** **Changes** **Comments** **Annotate** **Copy** **Actions** **Edit** **Recycle**

New subclasses have been created.

igneous rocks

colour

All instances **must** be classified.

Adamellite	<input checked="" type="radio"/> light	<input type="radio"/> dark
Andesite	<input checked="" type="radio"/> light	<input type="radio"/> dark
Basalt	<input type="radio"/> light	<input checked="" type="radio"/> dark
Dacite	<input checked="" type="radio"/> light	<input type="radio"/> dark

Below each rock type is a barcode-like graphic.

Figure 6.16: A completed form classifying each of the igneous rocks in terms of the new subclasses 'light' and 'dark'.

and is presented with the opportunity to create a subclass partition. Looking at the list of rocks, the first thing that comes into her head is to separate the rocks in terms of their shade, so she names the subclass partition 'colour', gives two possible groups, 'light' and 'dark' rocks. She checks the checkbox indicating that the subclass partition is exhaustive as all rocks are either light or dark. Figure 6.14 shows the completed form.

6.3.5 CLASSIFYING INDIVIDUALS

Once she has entered the details of the subclass partition, she clicks on the 'Classify' button and is presented with a page on which she can classify each of the rocks. She goes through the rocks one by one, but cannot remember the shades of some of the rocks, so just leaves them blank. Figure 6.16 shows the completed form. This method of generating subclass partitions is similar to a card sort (see Section 2.5.1.2).

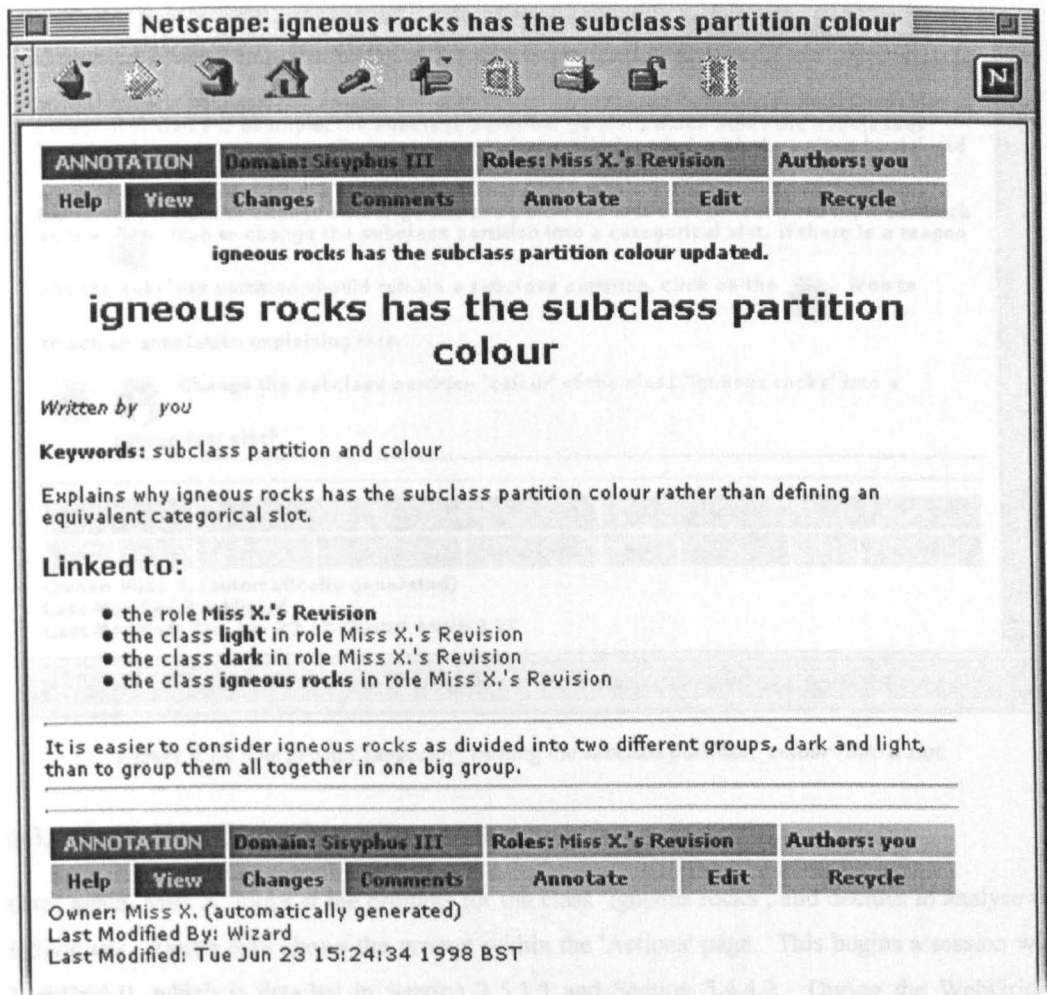


Figure 6.17: The annotation explaining Miss X.'s decision not to change the subclass partition 'colour' into a slot. The keywords; summary of the annotation; and links to her role and the classes 'igneous rocks', 'light' and 'dark' are created automatically, but she can change them or add more later.

6.3.6 PROVIDING EXPLANATIONS FOR IGNORED PROMPTS

Returning to the main view of her main class, 'igneous rocks', Miss X. can see that the rocks she has classified now appear within the class hierarchy as they should. Again, she clicks on the 'Actions' button to find out what to do next. This time, there is the suggestion that she changes the subclass partition 'colour' into a slot with the two values 'light' and 'dark'. Figure 6.18 shows the new prompt. She decides not to do this as she finds it easier to think of the rocks as the two separate groups, and writes an annotation instead explaining her decision (see Figure 6.17). This adds to the rationale for her ontology. Further detail about rationales and other discussion techniques are given in Section 6.5.

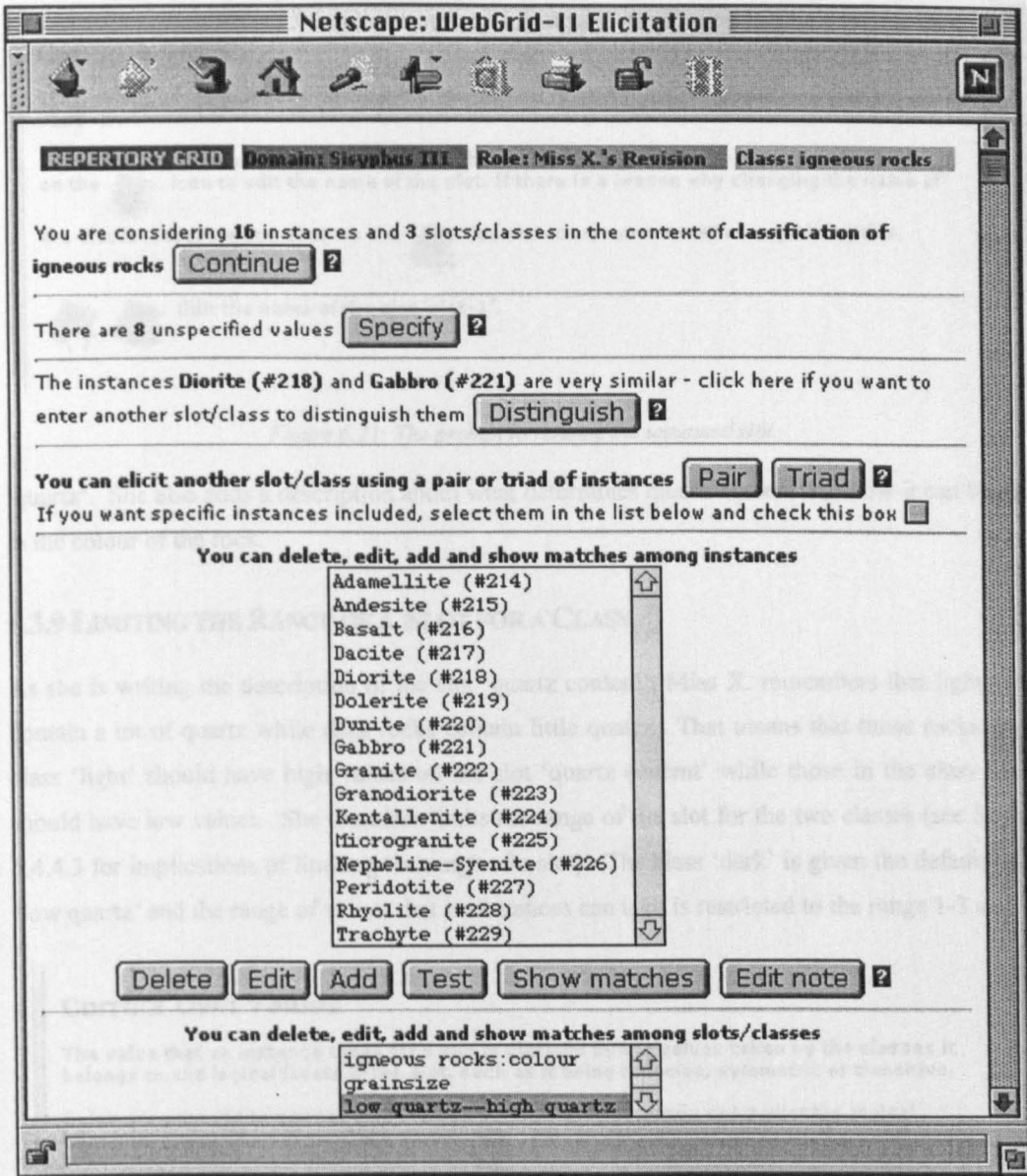


Figure 6.20: The WebGrid-II interface when used with APECKS.



session, Miss X. elicits two further constructs: the size of grains within a rock and the amount of quartz it contains. Figure 6.20 shows the WebGrid-II elicitation in progress.

6.3.8 CHANGING THE NAMES OF SLOTS

After a while, Miss X. returns to APECKS by clicking on the name of her role in the status bar (see Section 5.4.3.3) given at the top of the page in interactions with WebGrid-II. When she looks at the 'Actions' view, she finds that one of the constructs that she created within WebGrid-II has been translated into a slot but does not have a name (see Figure 6.21). She therefore renames it as

Rename Slots

If no names are specified for slots within the repertory grid, they get given the rather dull name slot-*n*.

Below are prompts to change the names of these slots into something more descriptive. Click on the  icon to edit the name of the slot. If there is a reason why changing the name of the slot is inappropriate, click on the  icon to attach an annotation explaining this.



  Edit the name of the slot 'slot-1'.

Figure 6.21: The prompt to rename the unnamed slot.


'quartz'. She also adds a description about what determines quartz content and how it can be seen in the colour of the rock.



6.3.9 LIMITING THE RANGE OF A SLOT FOR A CLASS

As she is writing the description of the slot 'quartz content', Miss X. remembers that light rocks contain a lot of quartz while dark rocks contain little quartz. That means that those rocks in the class 'light' should have high values on the slot 'quartz content' while those in the class 'dark' should have low values. She therefore limits the range of the slot for the two classes (see Section 5.4.4.3 for implications of limiting the range of a slot). The class 'dark' is given the default value 'low quartz' and the range of values that its instances can take is restricted to the range 1-3 and the

Correct Own Values

The value that an instance takes for a slot is dictated by the values taken by the classes it belongs to and logical facets of the slot, such as it being reflexive, symmetric or transitive.

Below are prompts to correct existing own values which currently contradict the logical constraints placed on them. Click on the  icon to get information about the constraints

and the  icon to edit the value of the slot for the frame. If there is a reason why the frame should take a contradictory value, click on the  icon to attach an annotation explaining this.




   The frame Andesite should not take the values 1 or "low quartz" for the slot quartz content. Edit the value?

Figure 6.22: A prompt indicating that an instance has a value for a slot that is logically inconsistent with the constraints set on the slot and the instance's types.

Class

☒ Update dark

Values definitely inherited by instances:
Up to 2 values can be taken.
Select from the following options:

low quartz
high quartz

Select from the following numbers:

1
2
3
4

Default values inherited by instances:
Up to 2 values can be taken.
Select from the following options:

low quartz
high quartz

Select from the following numbers:

1
2
3
4

Range:

Only numbers in the ranges:

☐ 1 to 5

☐ New range from 1 to 3

☐ The values: low quartz
high quartz

Same values as:

☐ grainsize

Cardinality:

☐ Single value slot

☐ A number of values between 0 and 2

Figure 6.23: Restricting the values for the class 'dark'. The default value is set to 'low quartz' and the range limited to the numbers 1-3 and the value 'low quartz'.

value 'low quartz'¹⁵. The default value and range for the class 'light' is similarly restricted. Figure 6.23 shows the completed form restricting the values of 'quartz content' for dark igneous rocks.

¹⁵ Rating-scale constructs from WebGrid-II are translated into slots that can take up to two values: one should be a number and the other, one of the labels of the extremes of the scale.

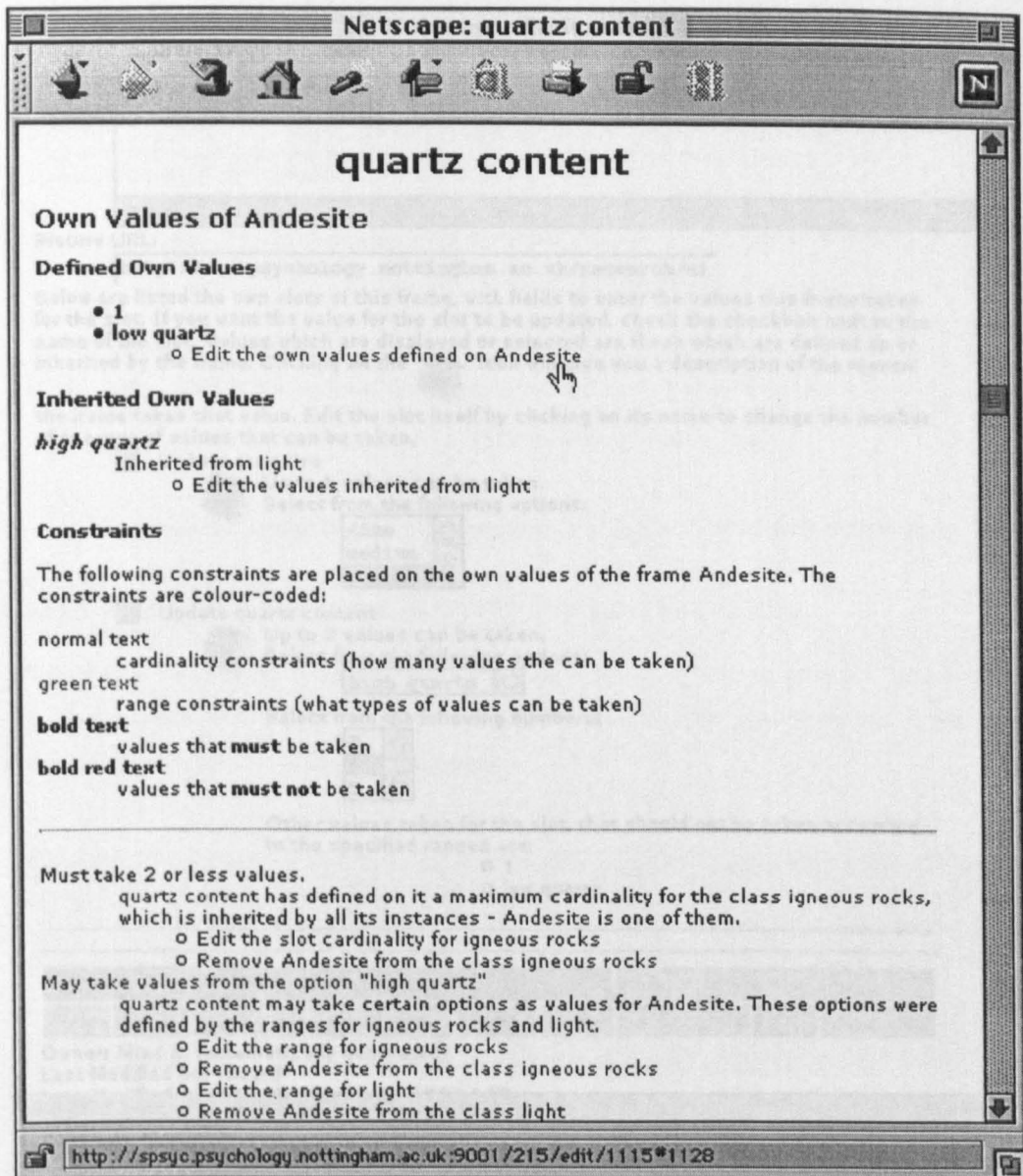


Figure 6.24: The constraints on the value of 'quartz content' for the individual 'Andesite' explained. Note that Miss X. can change any aspect contributing to the constraints placed on the value.

6.3.10 REDUCING INCONSISTENCIES

When Miss X. returns to the 'Actions' page, she finds that one of the rocks, Andesite, has an inconsistent value for 'quartz content' (see Figure 6.22). She clicks on the question-mark icon to find out why the value is inconsistent and is told that 'Andesite' has been classed as a light rock (which should have high quartz) but has been given low values for 'quartz content' (see Figure 6.24).

Section 5.4.4.3 explains in detail how consistency in slot values and class hierarchies are maintained within APECKS. In this case, Andesite has been assigned to the class 'light rock'

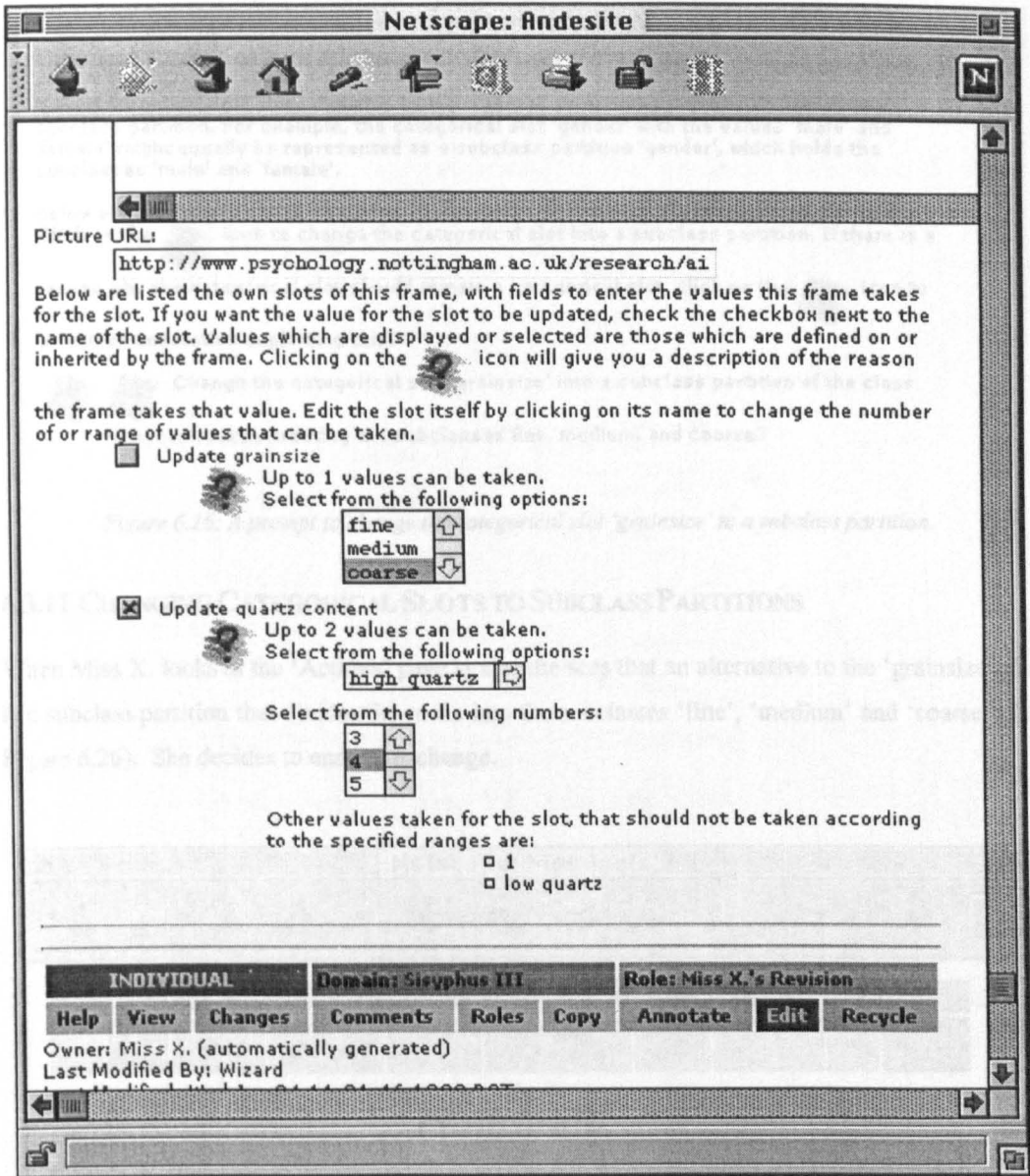




Figure 6.25: Changing the value of 'quartz content' for the individual 'Andesite'. Now that constraints have been set, only logically consistent values can be selected.

during the card sort shown in Sections 6.3.4 and 6.3.5. During the WebGrid-II interaction discussed in Section 6.3.7, it was given values of 1 and 'low quartz' for the slot 'quartz content'. In Section 6.3.9, however, the range of values taken by 'light' rocks was limited to numbers between 3 and 5, and the string 'high quartz'. Thus, Andesite has an inconsistent value for the slot 'quartz content': it should be in the range 3-5 and possibly include the string 'high quartz', but in fact it is 1 and includes the string 'low quartz'.

Miss X. decides that she was wrong when she said Andesite had low quartz content and right when she said it was a light rock. She goes to change the value (see Figure 6.25).

Change Categorical Slots to Subclass Partitions

Values for categorical slots can also be represented as membership of subclasses in a subclass partition. For example, the categorical slot 'gender' with the values 'male' and 'female' might equally be represented as a subclass partition 'gender', which holds the subclasses 'male' and 'female'.

Below are prompts to change existing categorical slots into subclass partitions instead. Click on the  icon to change the categorical slot into a subclass partition. If there is a reason why the categorical slot should remain a categorical slot, click on the  icon to attach an annotation explaining this.



  Change the categorical slot 'grainsize' into a subclass partition of the class 'igneous rocks', creating the subclasses fine, medium, and coarse?

Figure 6.26: A prompt to change the categorical slot 'grainsize' to a subclass partition.

6.3.11 CHANGING CATEGORICAL SLOTS TO SUBCLASS PARTITIONS

When Miss X. looks at the 'Actions' page again, she sees that an alternative to the 'grainsize' slot is a subclass partition that divides the rocks into the subclasses 'fine', 'medium' and 'coarse' (see Figure 6.26). She decides to enact this change.

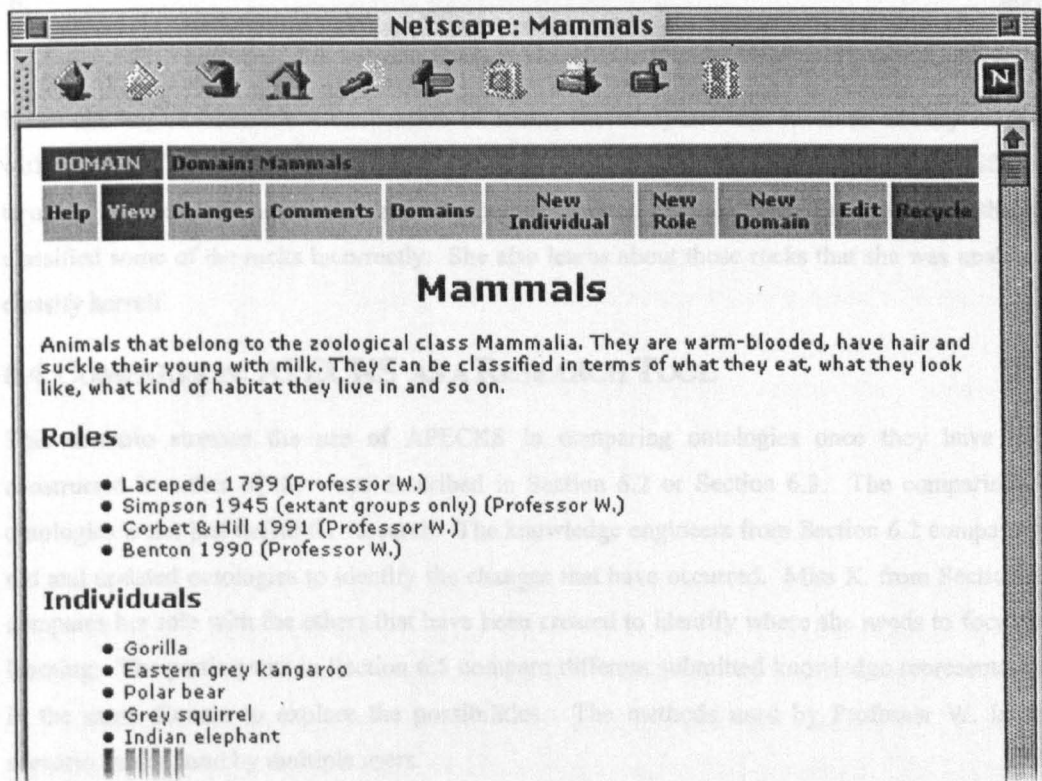


Figure 6.27: The 'Mammals' domain, showing the four roles created by Professor W..

Differing Individuals across Roles

The following pairs of roles do not contain the same individuals, on which comparison is based.

Simpson 1945 (extant groups only) and Lacepede 1799
The role Simpson 1945 (extant groups only) classifies different individuals than the role Lacepede 1799

Within Simpson 1945 (extant groups only)	Within both	Within Lacepede 1799
	Gorilla	
	Eastern grey kangaroo	
	Polar bear	
	Grey squirrel	
	Indian elephant	
	Sheep	
	Horse	
	Livingstone's flying fox	
	Sperm whale	
	Duck-billed platypus	none
	Western European hedgehog	
	Flying lemur	
	Three-toed sloth	
	Rabbit	
	Brown rat	
	Lion	
	Caribbean manatee	
	Sealion	
	American black bear	
Small-scaled tree pangolin		
Aardvark		
Rock hyrax		
Common tree shrew		
Rufous elephant shrew		

- Actions:
- Change the individuals in Simpson 1945 (extant groups only)
 - Change the individuals in Lacepede 1799

The rest of this comparison takes this discrepancy into account.

Figure 6.28: A summary of the different mammals classified in Simpson (1945) and Lacepede (1799).

When she has finished her classification of rocks, she compares this to those already existing within APECKS, using techniques described in Section 6.4. From this, she finds different terminology, such as ‘mesocratic’ and ‘leucocratic’ instead of ‘dark’ and ‘light’ and that she has classified some of the rocks incorrectly. She also learns about those rocks that she was unable to classify herself.

6.4 COMPARISON: APECKS AS A RESEARCH TOOL

This scenario stresses the use of APECKS in comparing ontologies once they have been constructed in either of the ways described in Section 6.2 or Section 6.3. The comparison of ontologies is not just useful for research. The knowledge engineers from Section 6.2 compare the old and updated ontologies to identify the changes that have occurred. Miss X. from Section 6.3 compares her role with the others that have been created to identify where she needs to focus her learning. The participants in Section 6.5 compare different submitted knowledge representations in the same domain to explore the possibilities. The methods used by Professor W. in this scenario can be used by multiple users.

Consensual Classes

The following classes have the same name and contain the same instances, showing a degree of **consensus** between the roles.

Mammalia

Within both the role Corbet & Hill 1991 (by Professor W.) and Simpson 1945 (extant groups only) (by Professor W.), the class Mammalia contains the following instances:

- o Gorilla
- o Eastern grey kangaroo
- o Polar bear

Within both the role Corbet & Hill 1991 (by Professor W.) and Simpson 1945 (extant groups only) (by Professor W.), the class Mammalia contains the following instances:

Within both the role Corbet & Hill 1991 (by Professor W.) and Simpson 1945 (extant groups only) (by Professor W.), the class Mammalia contains the following instances:

Within both the role Corbet & Hill 1991 (by Professor W.) and Simpson 1945 (extant groups only) (by Professor W.), the class Mammalia contains the following instances:

(extant groups only) and Corbet & Hill 1991

Rodentia

Within both the role Corbet & Hill 1991 (by Professor W.) and Simpson 1945 (extant groups only) (by Professor W.), the class Rodentia contains the following instances:

- o Grey squirrel
- o Brown rat

Actions:

- o Edit the name of Rodentia in Corbet & Hill 1991
- o Change the instances of Rodentia in Corbet & Hill 1991
- o Compare the classification of the instances of Rodentia under Corbet & Hill 1991 and Simpson 1945 (extant groups only)
- o Edit the name of Rodentia in Simpson 1945 (extant groups only)
- o Change the instances of Rodentia in Simpson 1945 (extant groups only)
- o Compare the classification of the instances of Rodentia under Simpson 1945 (extant groups only) and Corbet & Hill 1991

Lagomorpha

Within both the role Corbet & Hill 1991 (by Professor W.) and Simpson 1945 (extant groups only) (by Professor W.), the class Lagomorpha contains the following instances:

Within both the role Corbet & Hill 1991 (by Professor W.) and Simpson 1945 (extant groups only) (by Professor W.), the class Lagomorpha contains the following instances:

Figure 6.29: Extracts from the list of consensual classes from the comparison of Simpson's (1945) and Corbet & Hill's (1991) taxonomies.

The domain used within this scenario is that of 'Mammals', which was the domain used in the evaluation of APECKS discussed in Chapter 7. The roles generated within this scenario were used as 'gold standard' roles within the evaluation.

6.4.1 BACKGROUND

Professor W is interested in the history of mammalian taxonomies, particularly how convergent evolution has led to inaccurate classification only corrected today due to the development of molecular analysis. He decides it would be interesting to compare a number of taxonomies from different periods and see what changes have occurred.

6.4.2 CREATING MULTIPLE ROLES

Professor W first seeds the system (see Section 5.3.2 and Sections 6.2.2 and 6.2.3 for an example) by setting up a domain, 'Mammals', and selecting a representative sample of species from the various families of the currently accepted taxonomy. Under this, he creates a number of roles, each encoding a different taxonomy. He labels each taxonomy after its author and includes a short description that explains the situation in which the taxonomy was generated. The completed

Corresponding Classes

The following classes contain the same instances even though they have different names, showing a degree of **correspondence** between the roles.

Xenarthra (in Corbet & Hill 1991 by Professor W.) and Edentata (in Simpson 1945 (extant groups only) by Professor W.)

These classes both contain the instances:

- o Three-toed sloth

Actions:

- o Edit the name of Xenarthra
- o Create an annotation explaining why you have used the name 'Xenarthra'.
- o Change the instances of Xenarthra
- o Edit the name of Edentata
- o Create an annotation explaining why you have used the name 'Edentata'.
- o Change the instances of Edentata

Tubulidentata (in Corbet & Hill 1991 by Professor W.) and Tubilidentata (in Simpson 1945 (extant groups only) by Professor W.)

These classes both contain the instances:

- o Aardvark

Actions:

Figure 6.30: Extract from the list of corresponding classes from the comparison of Simpson's 1945 and Corbet & Hill's (1991) taxonomies. Note that the second in the list is simply a spelling mistake in the class name, and can be rectified by editing the name of the class.

domain is shown in Figure 6.27. He then sets about classifying mammals in a top-down fashion, using a combination of the techniques described in Section 6.2 and Section 6.3.

Conflicting Classes

The following classes have the same name but contain different instances, showing a degree of **conflict** between the roles.

Insectivora

Within Corbet & Hill 1991, Insectivora holds different instances to Insectivora within Simpson 1945 (extant groups only)

Within both

Within Simpson 1945 (extant groups only)

Western European hedgehog Rufous elephant shrew

Actions:

- o Change the instances of Insectivora in Corbet & Hill 1991
- o Create an annotation explaining why you have given these instances for 'Insectivora'.
- o Edit the name of Insectivora in Corbet & Hill 1991
- o Change the instances of Insectivora in Simpson 1945 (extant groups only)
- o Create an annotation explaining why you have given these instances for 'Insectivora'.
- o Edit the name of Insectivora in Simpson 1945 (extant groups only)

Primates

Within Corbet & Hill 1991, Primates holds different instances to Primates within Simpson 1945 (extant groups only)

Within both

Within Simpson 1945 (extant groups only)

Gorilla Common tree shrew

Actions:

Figure 6.31: Extract from the list of conflicting classes from the comparison of Simpson's (1945) and Corbet & Hill's (1991) taxonomies. Analysis techniques developed since 1945 have shown that the Rufous elephant shrew is not actually related to the Western European hedgehog despite their similarities and similarly that the Common tree shrew is not a primate.

Contrasting Classes

The following classes do not have the same name nor hold the same instances as any others, showing a degree of **contrast** between the roles.

Scandentia (in Corbet & Hill 1991 by Professor W.)

Does not share a name or instances with any classes within the roles under comparison.

Actions:

- Edit the name of Scandentia
- Change the instances of Scandentia
- Remove the class Scandentia (recycle it)
- Add a class named Scandentia to the role Simpson 1945 (extant groups only)

Pinnipedia (in Corbet & Hill 1991 by Professor W.)

Does not share a name or instances with any classes within the roles under comparison.

Actions:

- Add a class named Macroscelidea to the role Simpson 1945 (extant groups only)

Theria (in Simpson 1945 (extant groups only) by Professor W.)

Does not share a name or instances with any classes within the roles under comparison.

Actions:

- Edit the name of Theria
- Change the instances of Theria
- Remove the class Theria (recycle it)
- Add a class named Theria to the role Corbet & Hill 1991

Eutheria (in Simpson 1945 (extant groups only) by Professor W.)

Figure 6.32: Extract from the list of contrasting classes from the comparison of Simpson's (1945) and Corbet & Hill's (1991) taxonomies. Simpson's (1945) taxonomy used high-level distinctions between Theria, which have a womb, and Eutheria, which do not while Corbet & Hill's (1991) taxonomy focussed on families.

6.4.3 COMPARING CLASS HIERARCHIES

When Professor W. has completed the taxonomies, he looks at the comparisons between the class hierarchies (see Section 5.4.5). He does not want to change the taxonomies themselves, but uses the comparisons as a prompt for comments on the names of the classes used in the different periods, their origins and any reasons for changes made over time. Figure 6.28 to Figure 6.32 show various aspects of the comparison conducted by APECKS.

6.4.4 COMPARING ROLES USING WEBGRID-II

Professor W. then looks at the Sociogrid comparisons between the various roles using WebGrid (see Section 2.5.2.1 and Section 5.4.5). Figure 6.33 shows the page including the comparison. These comparisons highlight those mammals that are classified differently across the taxonomies, as can be seen from the extract in Figure 6.34. He notes down their names and returns to APECKS to add annotations explaining how new information about the mammals caused their classification to change over time.

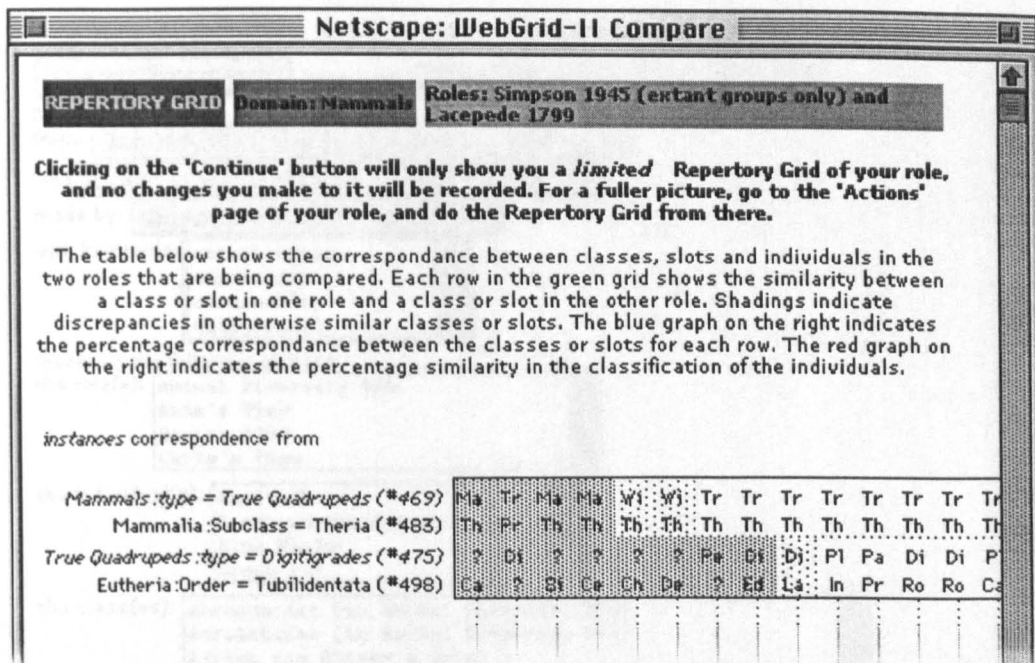


Figure 6.33: Page showing the WebGrid comparison between the Simpson's (1945) and Lacepede's (1799) taxonomies.

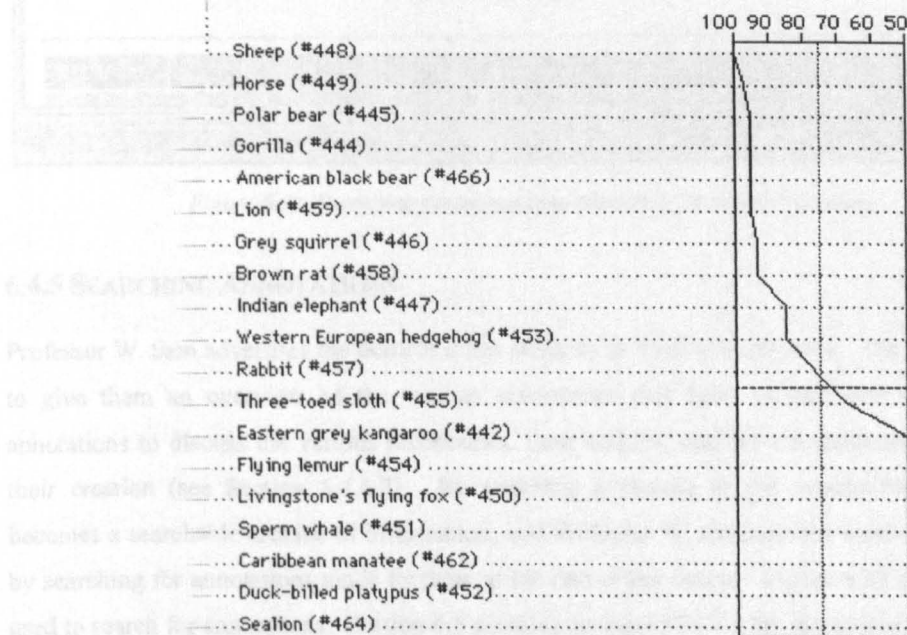


Figure 6.34: Extract from the WebGrid comparison between the Simpson's (1945) and Lacepede's (1799) taxonomies. There is high correspondence between the taxonomies in the classification of sheep and horses, but low correspondence for marine mammals.

search_comments Search for annotations and criteria on

Mammals

from: January 15 1998

to: June 24 1998

made by: anyone

with keywords: animal types
annotation
cardinality
categorical

involving the following objects:
the role(s): Animal Diversity Web
Anna's View
Benton 1990
Carly's View

the individual(s): Aardvark
American black bear
Blue Whale
Brown rat

the class(es): Abrocomidae (in Animal Diversity Web)
Acrobatidae (in Animal Diversity Web)
Africa (in Oliver's Role)
African (in Michael's View)

the slot(s): (in Anna's View)
diet (in Carly's View)
Intelligence (in Carly's View)
predators (in Caroline's View)

DOMAIN Domain: Mammals

Figure 6.35: Searching for annotations within the 'Mammals' domain.

6.4.5 SEARCHING ANNOTATIONS

Professor W. then advertises the domain to his students as a learning resource. The students use it to give them an overview of the various taxonomies that have existed over time, and use annotations to discuss the various taxonomies, their authors, and the circumstances surrounding their creation (see Section 5.4.6.3). By attaching keywords to the annotations, the domain becomes a searchable archive of information, and Professor W. monitors the input of his students by searching for annotations made by them at the end of the course. Figure 6.35 shows the form used to search for annotations. Section 6.5 discusses using APECKS for discussion in more detail.

6.5 DISCUSSION: APECKS AS A DECISION SUPPORT SYSTEM

This scenario stresses the use of APECKS for discussion (see Section 5.4.6), particularly for eliciting and using evaluative criteria in the creation of roles. It indicates the three contexts in which discussion takes place: the construction of evaluative criteria (Section 6.5.2 and 6.5.7); the building of a rationale (Section 6.5.4); and free annotation (Section 6.5.4).

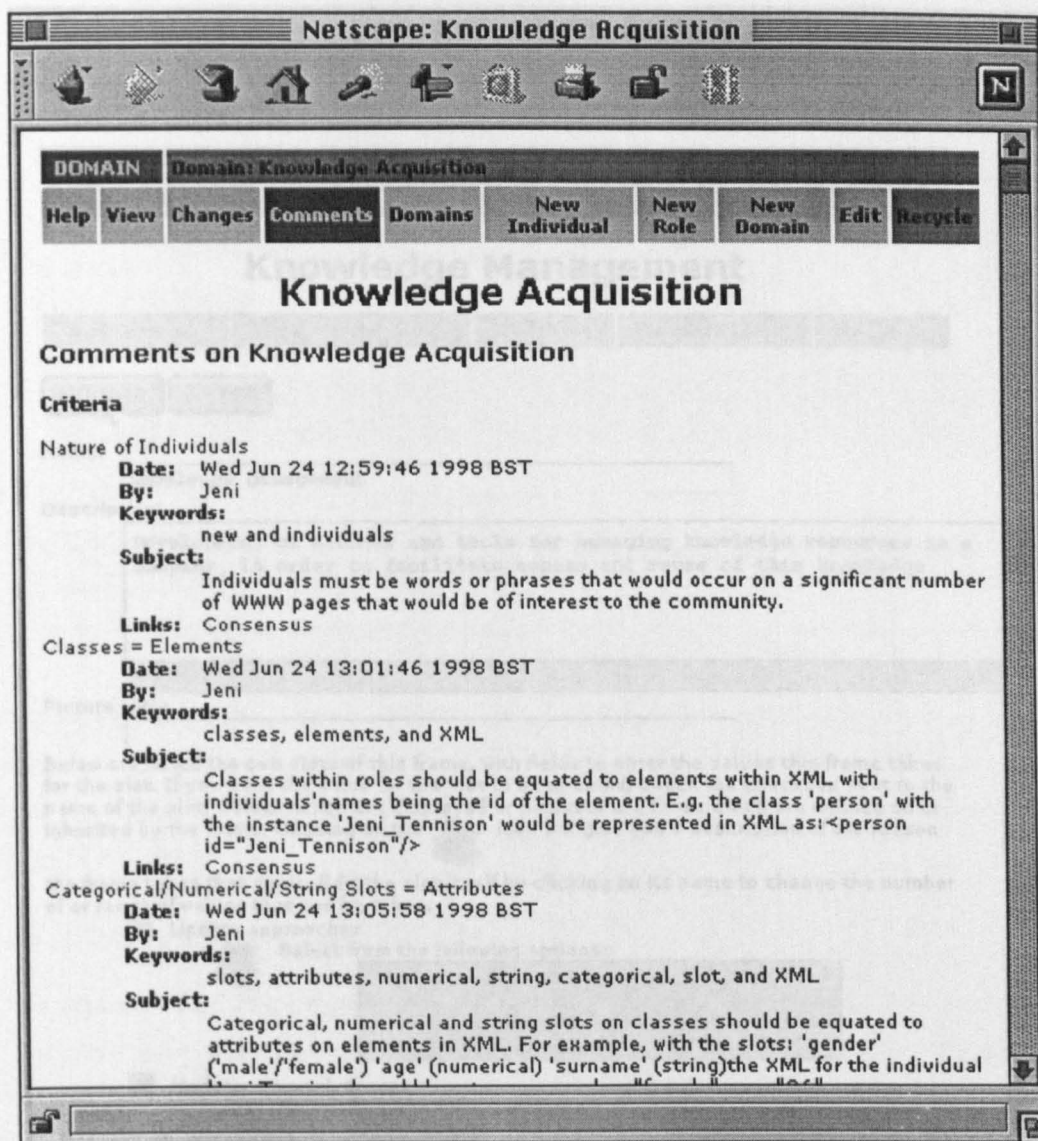


Figure 6.36: Criteria made within the domain 'Knowledge Acquisition'.

Discussion also occurs in the previous scenarios. In Section 6.2.8, knowledge engineers use the creation of a rationale to keep track of the changes and reasons for changes between an old and an adapted ontology. In Section 6.3.6, the rationale for the ontology constructed by a domain expert is added to through the creation of explanations for not acting on knowledge acquisition prompts. Finally, in Section 6.4.5, students use free annotation to discuss the differences between taxonomies based on their comparisons.

6.5.1 BACKGROUND

The KA community is interested in constructing an XML document type definition (DTD) that can be used to mark-up the pages of those within the community. This scenario mirrors the KA² initiative currently underway within the KA community (Benjamins & Fensel, 1998: see also

Netscape: Knowledge Management

INDIVIDUAL **Domain: Knowledge Acquisition** **Role: Consensus**

Help View Changes Comments Roles Copy Annotate Edit Recycle

Knowledge Management

Add Criterion Remove Criterion Add Class Remove Class Add Role

Update Reset

Name: Knowledge Management

Description: Development of methods and tools for managing knowledge resources in a company, in order to facilitate access and reuse of this knowledge.

Picture URL:

Below are listed the own slots of this frame, with fields to enter the values this frame takes for the slot. If you want the value for the slot to be updated, check the checkbox next to the name of the slot. Values which are displayed or selected are those which are defined on or inherited by the frame. Clicking on the ? icon will give you a description of the reason the frame takes that value. Edit the slot itself by clicking on its name to change the number of or range of values that can be taken.

☒ Update approaches ? Select from the following options:

- Knowledge Integration
- Knowledge Servers
- Enterprise Modelling
- Information Retrieval

☒ Update research groups ? Select from the following frames:

Figure 6.37: Entering information about Knowledge Management into the consensual role.

Section 2.5.2.3 and Section 8.5). While there already exist schemas that can be used to give information about people and publications, the community wants to have a formal encoding for both the research areas of people within the community, and the working relationships they have with each other.

6.5.2 CREATING CRITERIA

Before doing anything else, the organisers of the community effort come up with a number of rules to which those using the system are requested to comply. These can be used as evaluative criteria against which any representations that are generated can be assessed (see Section 5.4.6.1).

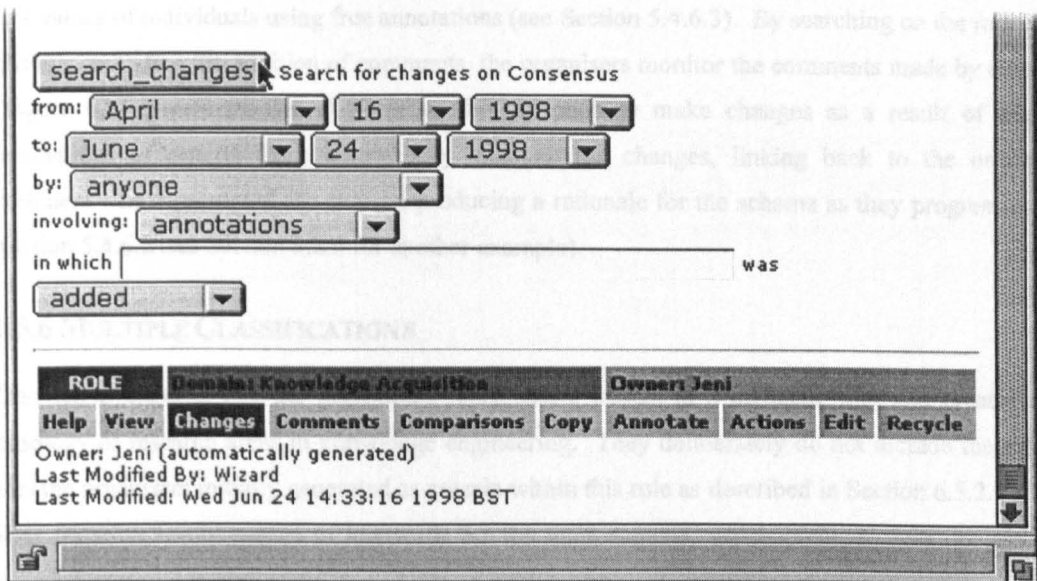


Figure 6.38: Searching for changes in the 'Knowledge Acquisition' domain. The search has been limited to those in which annotations were added between certain dates.

The rules are encoded as criteria within the domain and an annotation is attached to them stating that roles which do not comply to the criteria will be ignored when the decision is made on the final (or, rather, first-pass) schema. Figure 6.36 shows the list of criteria for the 'Knowledge Acquisition' domain.

6.5.3 CREATING 'CONSENSUAL' ROLES

The organisers then create a role that is designed to represent the consensus of the community. This role will be used as the basis of the final schema. In this they build their own suggested schema, using the techniques that are described in Section 6.2 and Section 6.3. They populate the role with individuals representing the major players in the community, people from their own research group, research groups they work with, publications they submit to and so on. They advertise this schema, inviting comments and alternatives from the rest of the community.

6.5.4 CREATING ADDITIONAL INDIVIDUALS

Graduate students, at the behest of their supervisors, start creating individuals representing their research groups into the domain and it gradually fills out with more information. The organisers fit this new information into their central schema. Figure 6.37 shows the information about the research topic 'Knowledge Management' being completed in the consensual role.

6.5.5 SEARCHING FOR CHANGES

Others in the community take time to investigate the suggested schema and start making comments on it, suggesting changes to the names of classes and slots or to the classification and

slot values of individuals using free annotations (see Section 5.4.6.3). By searching on the role for changes including the addition of comments, the organisers monitor the comments made by others (Figure 6.38 shows the search form used). When they make changes as a result of these annotations or criteria from others, they annotate the changes, linking back to the original comment which prompted the change, producing a rationale for the schema as they progress (see Section 5.4.6.2 and Section 6.2.8 for another example).

6.5.6 MULTIPLE CLASSIFICATIONS

One group starts to use the individuals representing papers in a different way and creates an ontology of research areas in knowledge engineering. They deliberately do not include the rules that the organisers initially generated as criteria within this role as described in Section 6.5.2. This later proves a useful source of keywords for the main schema, which includes slots describing research areas. It is also useful as a library of publications.

6.5.7 DISCUSSING DIFFERENCES BETWEEN ROLES

A few others in the community go so far as to create their own suggested schemas (i.e. they create separate roles that do abide by the rules generated in Section 6.5.2). By comparing these to the 'consensual' schema, the organisers can see what the changes are (see Section 5.4.5 and Section 6.4 for an example). In some cases, they discuss the pros and cons of the two methods with the originators of the other schemas, and often these discussions result in decisions that are encoded in criteria. Some are syntactic, such as "Class names should be in upper case", while others reflect attempts to limit the scope of the schema to knowledge engineering, such as "Natural Language Processing is not included in the schema."

The process continues for a month at which time the organisers decide to create the first version of the XML DTD for use by the rest of the community. Over that time, the schemas themselves have grown into a valuable resource of information for the community, especially those new to it.

6.6 CONCLUSIONS

This chapter has demonstrated the ways in which using a methodology based on a constructivist view of collaborative ontology construction could work in principle. It has illustrated the support given within APECKS for the main processes involved in collaborative ontology construction, as conceived in Section 5.3.2, namely:

- **Construction** of ontologies: Sections 6.2 and 6.3
- **Comparison** of ontologies: Section 6.4
- **Discussion** of ontologies: Section 6.5

This chapter has also demonstrated the flexibility of APECKS by showing its use in different contexts:

- **Ontology server** supporting the sharing and management of ontologies;
- **Learning tool** allowing novices to test and expand their expertise;
- **Research tool** providing automated comparison of classification structures;
- **Decision support system** aiding structured and focussed discussion.

Each of these contexts use the same features in slightly different ways and with slightly different goals. However, each of the tasks involved is, at heart, collaborative ontology construction: the taxonomies and schemas constructed in the latter scenarios are types of ontologies.

The chapter has also demonstrated some of APECKS's major features in action:

- Prompted knowledge elicitation from domain experts;
- Consistency checking;
- Use of WebGrid-II over the internet;
- Automated comparisons between classification hierarchies;
- Rationale creation;
- Searching for comments and changes;

Finally, this chapter has given some idea of the strengths and weaknesses of APECKS in terms of usability in selected scenarios by users familiar with the system. In the next chapter, I report on how novice users actually use APECKS and the degree to which the scenarios described here relate to real-life use.

CHAPTER 7 APECKS EVALUATION

7.1 SUMMARY

This chapter describes an evaluation study that was carried out to assess the extent to which APECKS supported the process of collaborative ontology construction. The study used twelve experts in the field of mammalian biology, who used APECKS for a total of 2½ hours each over a period of two and a half weeks. The study looked at three areas:

- The reported usability of the system.
- The quality of the ontologies produced by the subjects.
- The actions of the subjects using the system.

It found that APECKS could be successfully used by domain experts to construct personal ontologies within the domain. The knowledge acquisition support provided by APECKS (including WebGrid-II) and the comparisons between the subjects' roles were helpful to the subjects in creating these ontologies.

The study also identified three areas that need improvement:

- The presentation of information to users.
- The means of navigation through the system.
- The discussion system.

7.2 INTRODUCTION

APECKS is an implementation based on a methodology that is itself based on a theory about collaborative ontological engineering. The theory, based on a constructivist view of ontological engineering (see Section 2.3) states that collaborative activities need to be explicitly supported within ontology servers. It also claims that this support involves both maintaining diversity and encouraging discussion. The methodology suggests that this can be achieved by identifying differences and similarities between individuals' use of different concepts and terms (see Section 2.5.2.1). An evaluation of APECKS is therefore also, in part, an evaluation of the methodology and theory on which it is based.

APECKS is also an implementation that uses the World Wide Web (see Section 4.3.2), and makes use of networked resources (see Section 5.4.4.2 and 5.4.5). An evaluation of APECKS is therefore also, in part, an evaluation of a system that dynamically generates HTML. It is an evaluation of the utility of standard web browsers in accessing applications. And it is an evaluation of the practicalities involved in distributing applications across continents.

Finally, APECKS is an implementation that is intended for use by domain experts. An evaluation of APECKS is therefore also, in part, an evaluation of the idea that domain experts can construct semi-formal knowledge representations with only computer support (see Section 2.5.2.2).

As described above, APECKS is a system with many features. It is impossible to evaluate a single aspect separately from the others. The evaluation described here does not, and can not, give hard and fast findings about all the underlying issues, but it can provide a general picture of how APECKS is used in a real context.

As with the evaluation of any system, practical considerations limit the extent of the evaluation and its generality. These are discussed here as an explanation for some of the decisions made in the design of this evaluation study.

Firstly, there are few, if any, evaluations of other ontology servers that could serve as a baseline against which APECKS could be evaluated. Equally, it is difficult to carry out detailed evaluations of other systems as logging and automated quantitative evaluations of ontologies are not generally accessible. Without such comparisons to other systems, it cannot be said whether APECKS is better or worse than other ontology servers.

Secondly, there was the issue of finding people who would be willing to use the system within the budgetary constraints of the evaluation study. Although the final goal of APECKS is a system that can be used directly by domain experts, at the time of the evaluation many of the features that were designed to support this were not in place (see Section 8.3). On the other hand, trained knowledge engineers are both rare and busy: finding a sufficient number to use APECKS would have been impossible. For this reason, the subjects were undergraduate students.

Thirdly, there were multiple constraints on the choice of knowledge domain. The domain had to be one in which:

- Undergraduate students could be reasonably considered ‘expert’.
- Most, if not all, of the features of APECKS were used. For example, it needed to have a class structure and utilise slots.
- Sufficient disagreement, whether caused by differing opinion or expertise, would occur between subjects to make the comparisons and discussion worthwhile.
- Different tasks could be identified, to encourage differences between subjects.
- Individuals within the domain could be identified, and more added by the subjects.
- ‘Gold-standard’ ontologies existed to act as seeds for comparisons.

In the end, the domain of ‘mammals’ was decided upon, as it fitted most of the constraints given above. It should be noted that these constraints on domain do not apply to use of APECKS generally, but were desirable to increase the generality of the evaluation.

Fourthly, there was a limit on the realism of the use of APECKS. APECKS is designed for extended use, but there was a limit to the amount of time subjects would spend using the system. APECKS is also designed for concurrent and overlapping use, with people using it at the same time as others, as well as before and afterwards. However, in order to monitor their use of the system, subjects had to have separate sessions, and in order for them to achieve anything during these sessions, they had to be of a reasonable length. These issues were resolved by having four sessions per subject, distributed over a three-week period.

7.3 METHOD

7.3.1 SUBJECTS

Twelve subjects completed the study. They were volunteers, paid £10 for their participation. All the subjects were undergraduates in the School of Life Science at the University of Nottingham.

7.3.2 DESIGN

The subjects were divided randomly into three groups. Unfortunately, subject dropout resulted in unequal groups. Following consultation with a domain expert, three possible purposes for the collection of information from the subjects were generated. These were:

- **morphology:** "We are building a computer-based 'Mammal-Spotter's Guide' that will help people identify any mammals they see."
- **diet:** "We are building a computer-based guide for zoos about what to feed the mammals in their care."
- **habitat:** "We are building a computer-based tour guide that will help people learn about what mammals they are likely to encounter when they go on holiday."

One of these tasks was assigned to each group: subjects within a particular group received differing instructions (as given above) depending on the task assigned to that group.

7.3.3 MATERIALS

Four questionnaires were used during the course of the study. The subjects were given a background questionnaire at the beginning of the experiment (see Appendix II). This questionnaire asked the subjects about the extent of their experience with computers and their knowledge about mammals.

At the end of each session using APECKS, the subjects were given a usability questionnaire (see Appendix III). This questionnaire highlighted six activities: browsing; adding, changing and removing information (see Section 5.4.4); comparing roles (see Section 5.4.5) and discussing information (see Section 5.4.6). The subjects were asked to rate how easy each activity was on a seven-point Likert-like scale ranging from 'Extremely difficult' (1) to 'Extremely easy' (7). They were also given space to make comments about the easiest and most difficult aspects of using the system, and for general comments. The subjects were asked to estimate the ease of carrying out those tasks they had not actually attempted.

On completion of the final session, the subjects were asked to give, in their own words, definitions for the various knowledge engineering terms used within APECKS.

The final questionnaire (see Appendix IV) was completed by a knowledge engineer and assessed the ontologies that were produced by the subjects. The ontologies were rated on seven-point Likert-like scales in terms of detail, precision, breadth, consistency, completeness, readability, the extent to which the ontology had been discussed, and its utility. This questionnaire also asked the knowledge engineer to indicate which aspects of knowledge about mammals were addressed.

The subjects were given a vocabulary sheet (see Appendix V) with definitions and examples of the terms 'annotation', 'cardinality', 'class', 'comparison', 'criterion', 'default value', 'domain', 'individual', 'inverse', 'range', 'repertory grid', 'representation', 'role', 'slot', 'subclass partition' and 'value'. They were allowed to refer to the sheet throughout the sessions.

There were two sets of instruction materials (see Appendix VI and Appendix VII), each of which had three variations, for the three tasks. The first set of instructions asked the subjects to read the definitions of the terms 'role', 'individual', 'class' and 'slot', and drew the subjects' attention to the status bar and button bar, specifically to the 'Help' and 'Actions' buttons. The second set of instructions asked the subjects to read the definitions of the terms 'comparison', 'annotation' and 'criterion' and drew the subjects' attention to the 'Comparisons', 'Comments' and 'Annotate' buttons. The second set also explained that other people had used the system and encouraged the subjects to look at the comparisons and make comments on other people's roles. Both sets of instruction materials asked the subjects to try to make any information they added to the system as clear as possible and informed them that their actions would be logged. They also both requested them to avoid using the 'Back' button on the web browser, which can lead to problems of forms being resubmitted.

Twenty individuals (see Appendix VIII) and five roles (see Appendix IX to Appendix XIII) were chosen by a domain expert and entered into APECKS as a seed set for the study (see Section 5.3.3.1). The individuals were all *species* of mammals, and were chosen such that each of the classes in the seed roles contained at least one individual. This was to ensure that there was variety in the individuals to encourage broad roles. Three of the roles were very similar genetically based taxonomies, including only extant groups. They differed in the depth of the hierarchy they used and, in a few cases, in the names used for the classes. Of the two other roles, one was based on Lacepede's (1799) taxonomy, which classifies mammals in terms of their physical characteristics, and the other classified mammals in terms of the zoogeographical region in which they can be found. These roles were intended as illustrative examples against which the subjects could compare their own roles.

7.3.4 EQUIPMENT

The APECKS server ran on a Sparc Server 20/51 with 224Mb RAM running under Solaris 2.5. The subjects used Netscape Navigator 3.01 running on a Power Macintosh 8100/100AV with 12Mb RAM in order to access the APECKS server.

7.3.5 PROCEDURE

The subjects attended four sessions over a period of two and a half weeks. During the first week, each subject attended an introductory session that lasted for an hour. They first completed the background questionnaire and read the first set of instruction materials. Following this, the experimenter led them through the first steps in creating a role. They were first shown an example role in a different domain, to give them an idea about the kind of representation that they would be creating. They were then shown how to access their own role and how to access a list of prompts for further actions through the 'Actions' button. The experimenter led them through the selection

of individuals for the role, recommending that they selected around 15 from the list; the creation of a top-level class called 'mammals'; and the addition of individuals to the class. They were then left to continue creating the role on their own. Actions taken while the experimenter was helping in these initial stages were not included in the data analysed below. This introduction took around 20 minutes. Ten minutes before the end of the session, the subjects were stopped (although they were permitted to complete and submit any forms they were filling in) and asked to complete a usability questionnaire.

The rest of the sessions lasted for half an hour each. At the beginning of each session, the subjects read the second set of instruction materials; at the end of each session, they completed the usability questionnaire. At the beginning of the second session, the experimenter led the subjects through the process of comparing roles with each other, and showed the subjects how to return to their role after doing so. At the end of the final session, the subjects were given a final questionnaire to assess their understanding of the knowledge engineering terminology used within APECKS, along with the usability questionnaire. They were then given £10 for completing the experiment.

Throughout the sessions, the system logged the APECKS pages that were visited and the length of time spent at each. At the end of each session, the system recorded measures of the subject's ontology's state, namely: the number of each type of object; the number of hierarchies present within the ontology; and the number of subclass partitions that had been created.

7.4 RESULTS

7.4.1 SYSTEM USABILITY

7.4.1.1 Time Series Analysis

Time series analysis of the usability ratings given over the four sessions were carried out to see whether the system became easier to use as time went on. Figure 7.1 shows how the mean rating of ease of each of the activities changed over the sessions.

Finding Information

The rating of the ease of finding information during the third session was significantly and moderately negatively skewed, and the ratings for finding information were thus transformed. A one-way within subjects ANOVA showed that the ratings of ease of finding information differed significantly session by session ($F(3, 33) = 8.93$, $MS_{\text{error}} = 0.07$, $p < 0.001$). Paired-sample t-tests showed the ratings were significantly higher in the final session than they were in the first ($t(11) = 3.40$, $p < 0.01$). The improvement was particularly marked between the second and third sessions ($t(11) = 5.91$, $p < 0.01$).

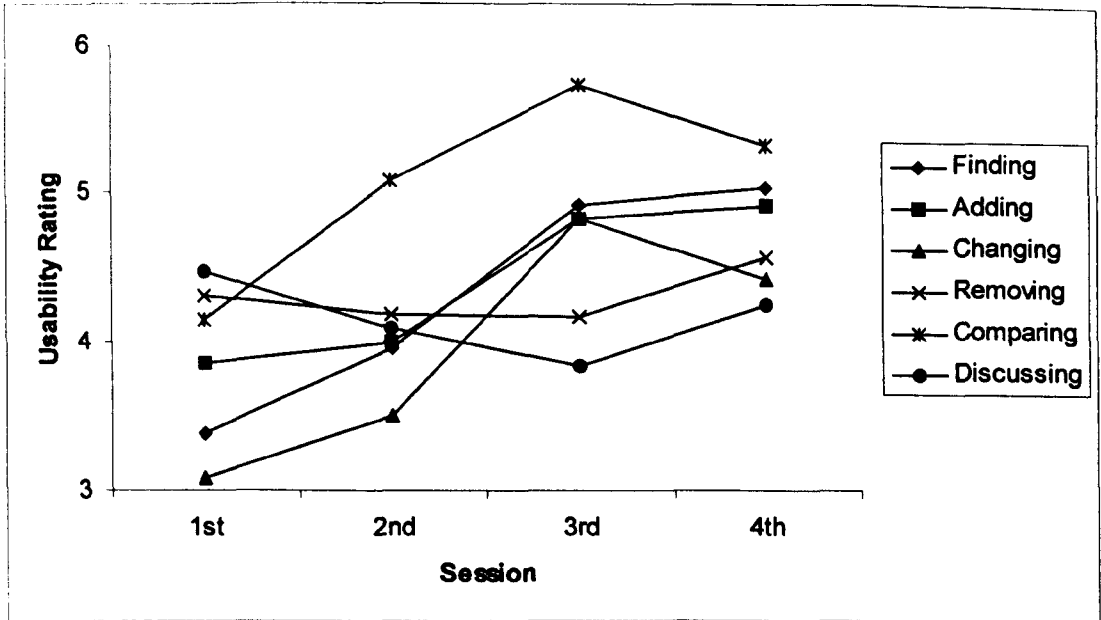


Figure 7.1: Mean rating of ease of carrying out activities over the four sessions.

Adding Information

A one-way within subjects ANOVA showed that the ratings of ease of adding information differed significantly session by session ($F(3, 33) = 4.18$, $MS_{\text{error}} = 0.89$, $p < 0.05$). Paired-sample t-tests showed the ratings were significantly higher in the final session than they were in the first ($t(11) = -2.31$, $p < 0.05$). The improvement was particularly marked between the second and third sessions ($t(11) = -4.02$, $p < 0.01$).

Changing Information

A one-way within subjects ANOVA showed that the ratings of ease of changing information differed significantly session by session ($F(3, 33) = 3.19$, $MS_{\text{error}} = 2.27$, $p < 0.05$). There was no significant improvement in the ratings given on the final session over those given on the first session ($t(11) = -1.50$, $p > 0.05$), but paired-sample t-tests showed a significant improvement between the second and third sessions ($t(11) = -4.00$, $p < 0.01$).

Removing Information

A one-way within subjects analysis of variance showed that the ratings of ease of removing information did not differ significantly session by session ($F(3, 30) = 0.49$, $MS_{\text{error}} = 1.60$, $p > 0.05$).

Comparing Roles

A one-way within subjects analysis of variance showed that the ratings of ease of comparing roles differed significantly session by session ($F(3, 30) = 4.07$, $MS_{\text{error}} = 1.11$, $p < 0.05$). Paired sample

	First	Second	Third
Second	.02		
Third	.32	.91	
Fourth	.29	.81	.91

Table 7.1: Correlations between usability over sessions (bold indicates a significant correlation).

t-tests then showed a significant difference between the ratings given on the first and final sessions ($t(11) = -2.92, p < 0.05$), with no significant differences in the ratings from session to session.

Discussing Information

A one-way within subjects analysis of variance showed that the ratings of ease of discussion did not differ significantly session by session ($F(3, 30) = 2.19, MS_{error} = 1.11, p > 0.05$).

7.4.1.2 Usability Scale

The usability ratings of the six areas (finding, adding, changing & removing information, and comparing and discussing roles) were highly intercorrelated, and were therefore summed into a single usability scale for the rest of the analysis. The generated scales for the four sessions had good internal reliability ratings of between 0.77 (for the first session) and 0.92 (for the third session) and a possible range from 6 to 42.

The general usability reported by the subjects during the first session was not significantly correlated with that reported on later sessions, whereas that reported during the second, third and fourth sessions were all highly intercorrelated, as shown in Table 7.1.

There was little relation between subjects' experience with computers and the usability ratings they gave for APECKS. During the first session, there was a significant **negative** correlation between subjects' experience with the WWW and their rating of the ease of use of APECKS ($r = -0.64, p < 0.05$). This indicated that those with less experience with the WWW found the system easier to use. During the second session, when the subjects were first introduced to annotations, there was a significant positive correlation between subjects' reported experience with email and their rating of the ease of use of APECKS ($r = 0.64, p < 0.05$). This indicated that those with more experience with email found using APECKS easier during the second session. These differences in usability depending on experience disappeared in later sessions.

7.4.1.3 Comments

The comments given on the questionnaires highlighted a number of areas that were not addressed specifically by the questionnaire.

Presentation

There were several comments about the difficulty of the terminology. There were several comments on terminology that was MOO-specific (such as using 'Recycle' instead of 'Delete') or CSCW-oriented (such as using 'Annotate' rather than 'Comment'). The use of knowledge engineering terminology was a particular problem:

The words role, class, domain etc etc are all very confusing despite having the vocab sheet to explain them.

The layout of the information and the size of the text used caused some problems: one subject in particular found the text too small to be read easily. The formal way in which the pages were structured was particularly liked by some:

The information is presented in an easy to follow style, like the keys we had to make in GCSE biology.

The subjects reported liking the ease with which they could focus on a particular object or aspect of their role simply by clicking on its name. They also liked being able to change that aspect without worrying about the rest of the role:

You can get information at different levels in the roles i.e. you can find out about the class or the individual without having to get involved with the other.

There were also several requests for a general overview of the role to be on screen at all times.

Navigation

The subjects also complained that they got lost easily, both in terms of not being able to work out where they were, and in terms of not being able to find their way to where they wanted to go. In these situations, the main page of their own role and the main page of the domain acted as landmarks:

Whatever happens "Caroline's Role" [the main page of the subjects' role] gets me back to a familiar page.

One subject suggested that a button should be available at all times to allow users to return to the main page of their own role.

The comments did show that as the subjects used the system more they found it easier to navigate:

Finding information is easier after each session when I am more familiar with the programme.

The subjects reported finding the limitation of not being able to use the web browser's 'Back' button particularly annoying. Without the 'Back' button, subjects sometimes had to go through several other pages before returning to where they had been:

When you move on a step but then want to go back, you can't always do it without going right back to the beginning.

The subjects also found it difficult to correct any errors they made, particularly as there was no way to go back or undo what they had just done, and because they found it hard to navigate around the system.

I wrote some information as a criterion and then realized I had made a minor mistake but could not work out how to get back to it and amend it.

There were a few complaints about time delays between selecting actions and the page being displayed. This was a particular problem with WebGrid-II: the display of pages is slower due to the distance between the APECKS server and the WebGrid-II server:

[What was difficult about finding information?] Time taken for computer to think about what you want it to do – because you don't know if it is a problem or not.

Discussion

The subjects were not enamoured of the discussion system. The subjects reported not having anything to reply to and were reticent to comment on others' roles directly. The fact that discussion was asynchronous and that they could not receive replies immediately was also frowned upon. On the other hand, they found that this made it easier to structure their comments:

You have time to think and formulate your argument, and it is therefore better constructed and laid out.

Ontology Construction and Comparison

The subjects reported that the prompts giving suggested actions were very useful and liked having a lot of choice about what to do next. However, a few were overwhelmed with the number of choices with which they were presented:

As with adding information, I was unsure about what I should be doing. There were too many options once you decided upon an action.

There was particular approval for the comparisons between roles. Subjects reported using them for three main purposes:

- To give new ideas for the development of their own role:

[What was easy about comparing roles?] Straight-forward + easy to follow. Good at providing new ideas and helping develop 'Anna's View' [the subject's own role].

- To act as a jumping-off point from which they could explore someone else's role:

The break down of all the differences + similarities is very clear and means you can choose easily which one to investigate.

- To give ideas for comments they could make on others' roles:

Nice set up – the screen shows exactly what the similarity and differences are so you know clearly what to comment on.

Some subjects thought that the roles were too different from each other to make the comparisons useful, or that they had equal validity and therefore did not feel comfortable making comments on them:

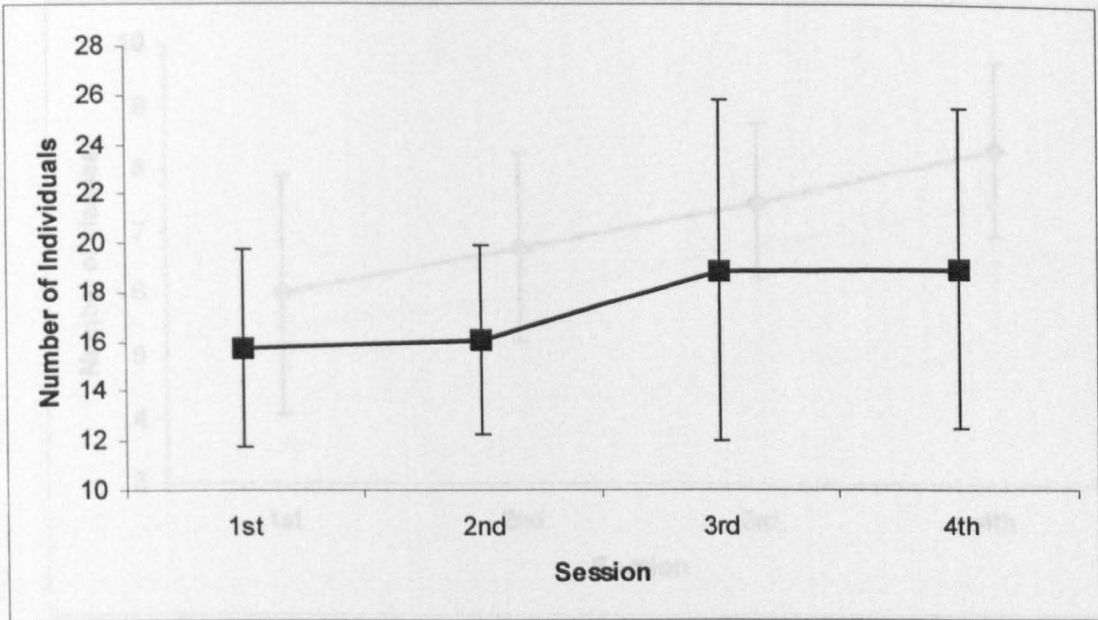


Figure 7.2: Number of individuals cumulatively over sessions. Bars indicate one standard deviation.

The comparison was not very worthwhile, often people had used a totally different method of categorising which was totally justified, or used the same idea with slightly different words/categories.

There were a couple of suggestions for improvements to the comparisons. One subject suggested that the system should count terms that were almost identical (such as the singular and plural of a noun) as the same term. Another suggested that the system should give ideas for why the differences between roles might occur.

7.4.2 ONTOLOGY QUALITIES

7.4.2.1 Quantitative Measures

Various features of the subject's role were recorded at the end of each session. These features were:

- Number of **individuals** within the role
- Number of **classes** within the role
- Number of **slots** within the role

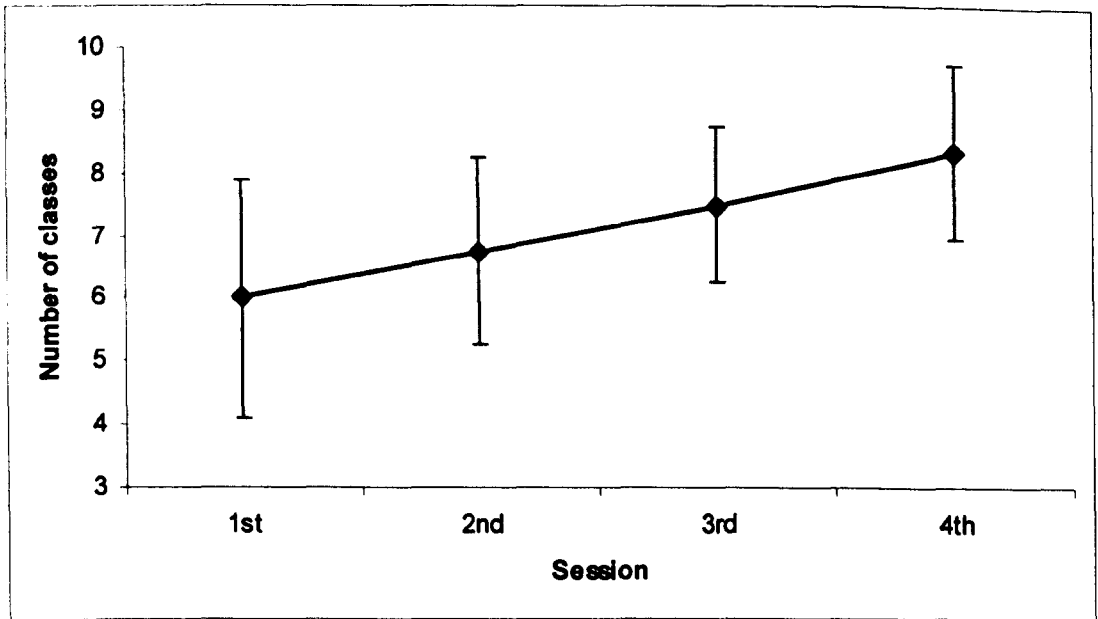


Figure 7.3: Number of classes cumulatively over sessions. Bars indicate one standard deviation.

- Number of distinct **hierarchies** within the role¹⁶
- Number of named **subclass partitions** within the role
- Number of **annotations** within the role
- Number of **criteria** within the role

The quantitative measures taken after each session were examined to see if the makeup of the ontologies altered over time.

Individuals

A one-way within-subjects ANOVA showed that the number of individuals within the roles changed significantly over the sessions ($F(3, 33) = 3.31$, $MS_{\text{error}} = 11.24$, $p < 0.05$). The increase between the first and final sessions approached significance ($t(11) = -2.97$, $p < 0.10$), but there were no significant differences from session to session. Figure 7.2 shows the changes over the sessions.

Classes

A one-way within-subjects ANOVA showed that the number of classes within the roles changed significantly over the sessions ($F(3, 33) = 9.52$, $MS_{\text{error}} = 1.26$, $p < 0.001$). There was a significant increase in the number of classes in the final session over the first session ($t(11) = -3.24$, $p < 0.01$),

¹⁶ The number of hierarchies within a role was calculated as the number of distinct trees below the root class. It thus indicates the number of distinct ways in which the individuals are classified within the role.

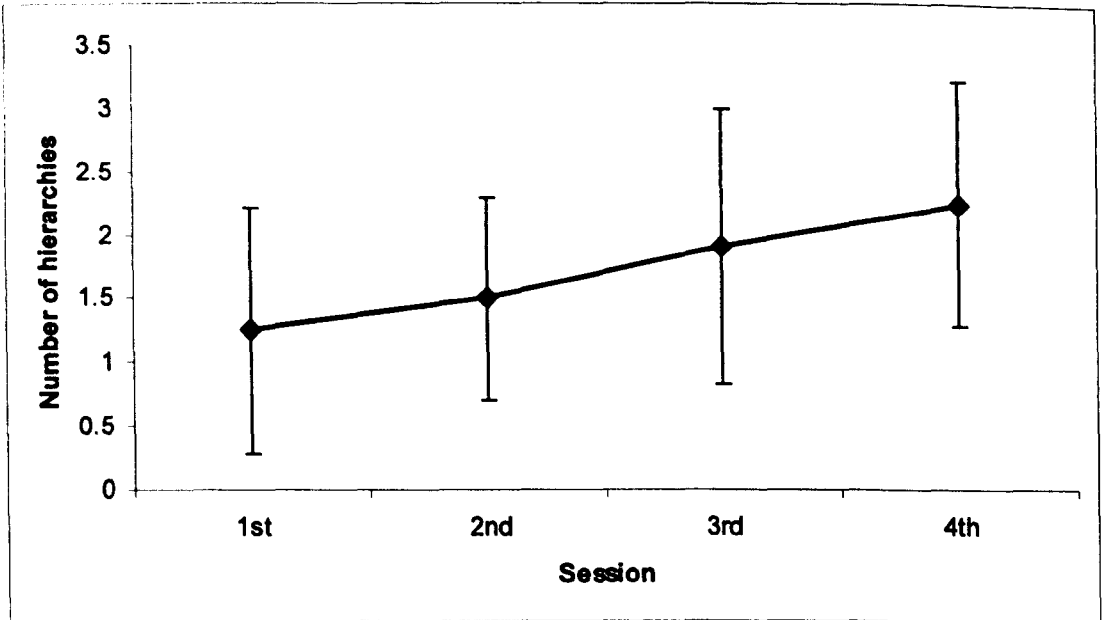


Figure 7.4: Number of hierarchies cumulatively over sessions. Bars indicate one standard deviation.

and significant increases from session to session: between the first and second ($t(11) = -2.69$, $p < 0.05$), second and third ($t(11) = -2.46$, $p < 0.05$) and third and final ($t(11) = -2.42$, $p < 0.05$). Figure 7.3 shows the increase in classes over the sessions.

Slots

There was no significant change in the number of slots on the subjects' roles over the sessions ($F(3, 33) = 1.63$, $MS_{\text{error}} = 0.58$, $p > 0.05$).

Hierarchies

A one-way within-subjects ANOVA showed that the number of hierarchies represented within the roles changed significantly over the sessions ($F(3, 33) = 5.89$, $MS_{\text{error}} = 0.40$, $p < 0.01$). There was a significant increase in the number of hierarchies between the first and final sessions ($t(11) = -2.71$, $p < 0.05$), but not between any of the individual sessions. Figure 7.4 shows the increase in hierarchies per role over the sessions.

Subclass Partitions

There was no significant change in the number of subclass partitions on the subjects' roles over the sessions ($F(3, 33) = 0.79$, $MS_{\text{error}} = 0.07$, $p > 0.05$).

Annotations

A one-way within-subjects ANOVA showed that the number of annotations on the roles changed significantly over the sessions ($F(3, 33) = 11.15$, $MS_{\text{error}} = 0.47$, $p < 0.001$). There was a significant increase in the number of annotations given on the roles between the first and final

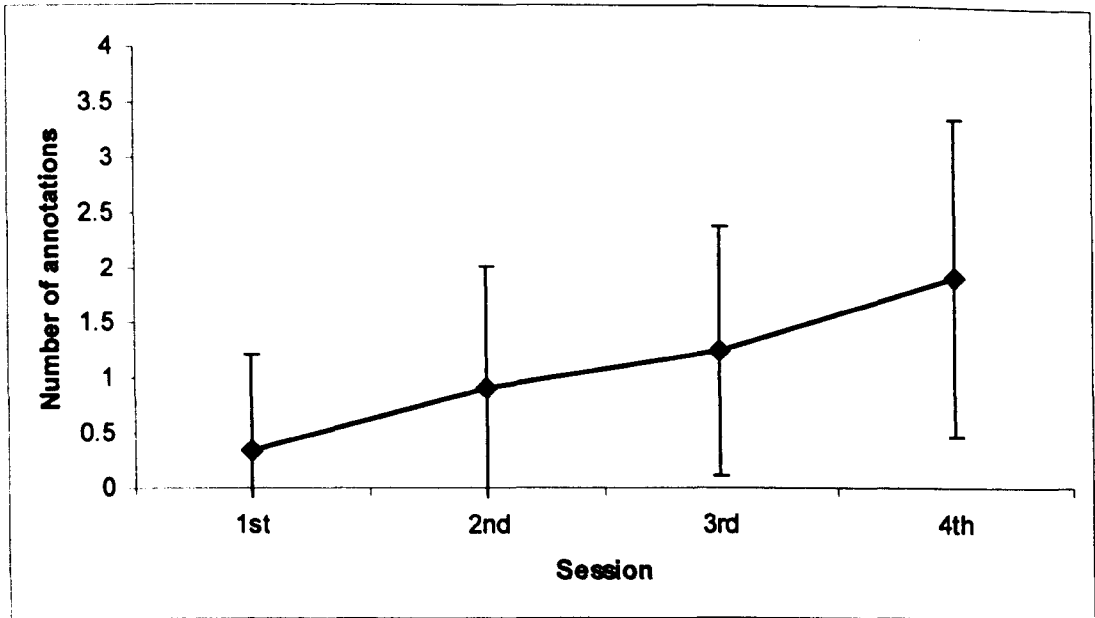


Figure 7.5: Number of annotations cumulatively over sessions. Bars indicate one standard deviation.

sessions ($t(11) = -3.64$, $p < 0.01$), and significant increases from session to session: between the first and second ($t(11) = -2.24$, $p < 0.05$), the second and third ($t(11) = -2.35$, $p < 0.05$), and between the third and final sessions ($t(11) = -2.97$, $p < 0.05$). Figure 7.5 shows the mean number of annotations made on each role over the sessions.

Criteria

There was no significant change in the number of criteria on the roles over the sessions ($F(3, 33) = 1.00$, $MS_{\text{error}} = 0.02$, $p > 0.05$).

7.4.2.2 Qualitative Scale

The quality of the ontologies produced by the subjects, and those used as seeds, was assessed at the end of the experiment. The dimensions on which the ontologies were rated were highly correlated, for the most part, as can be seen in Table 7.2, with the exception of the rating for the ontology being a source of discussion. The other ratings were summed to give an overall scale of 'qualitative ontology goodness' with an internal reliability of 0.89 and a possible range from 7 to 49.

7.4.2.3 Quantitative & Qualitative Correlations

The correlations between the quantitative measures of the final ontologies and the qualitative assessment were examined. There was a significant positive correlation between the number of individuals within a role and its qualitative goodness ($r(18) = 0.59$, $p < 0.05$) and between the number of classes within a role and its qualitative goodness ($r(18) = 0.65$, $p < 0.01$). There was

	Detail	Precision	Breadth	Consistency	Completeness	Readability	Discussion	Utility
Detail	1.00							
Precision	.72	1.00						
Breadth	.90	.70	1.00					
Consistency	.09	.50	.11	1.00				
Completeness	.81	.86	.83	.34	1.00			
Readability	.04	.50	.08	.67	.31	1.00		
Discussion	-.17	-.28	-.04	-.26	.29	.11	1.00	
Utility	.72	.85	.68	.46	.86	.43	-.32	1.00

Table 7.2: Correlations between qualitative ontology scales (bold indicates a significant correlation).

also a significant negative correlation between the number of annotations on a role and its qualitative goodness ($r(18) = -0.58, p < 0.05$).

7.4.2.4 Seed Ontologies & Subjects' Ontologies

Unsurprisingly, the seed ontologies were judged to be significantly better on the qualitative scale ($t(16) = 6.84, p < 0.001$), with a mean rating of 42.8 compared to the subjects' ontologies mean rating of 23.4. The seed roles had significantly fewer annotations on them (a mean of 0.3) than the subjects' roles (a mean of 1.9) ($t(16) = -2.57, p < 0.05$). They also had significantly fewer slots ($t(11) = -2.35, p < 0.05$), as none were given to them whereas subjects added an average of 1.7 to their roles. The seed and subjects' ontologies did not otherwise differ quantitatively.

7.4.2.5 Task Effects

A one-way ANOVA was carried out to confirm that the qualitative goodness of the ontologies did not differ across the three task groups. It was not significant at the 5% level of significance ($F(2, 9) = 1.59, MS_{\text{error}} = 38.65, p > 0.05$).

A MANOVA was performed to examine the effect of the task group on the coverage of different areas of mammalian knowledge. There was no significant multivariate effect on the degree to which the ontologies addressed mammalian morphology, diet and habitat. However, there was a significant univariate effect on the degree to which mammalian morphology was addressed ($F(2, 9) = 6.09, MS_{\text{error}} = 1.99, p < 0.05$). Between-subjects t-tests revealed that those subjects who had been told that the knowledge would be used for a mammal-spotter's guide addressed mammalian morphology significantly more than those who had been told it would be used for dietary advice

for zoos ($t(7) = 3.25$, $p < 0.05$), with an average rating of 5.3 compared to 2.4 on the 1-7 scale. There were no significant differences on the amount morphology was addressed between the 'morphology' and 'habitat' groups, nor between the 'diet' and 'habitat' groups.

A chi-square test revealed no significant association between the task subjects were asked to perform and the most addressed area as reported by the knowledge engineer ($\chi^2(2) = 2.88$, $p > 0.05$).

7.4.2.6 Background Effects

There were no significant correlations between the experience the subjects had with email, the WWW or MOOs and the qualitative goodness of the ontologies they produced or the quantitative measures of the makeup of the roles that were produced. There was also no significant correlation between the quality of the ontologies and the subject's reported experience with mammals. The reported usability of the system during each of the sessions was also not significantly correlated with the quality of the ontologies produced.

MANOVAs and chi-square tests revealed no significant effects of the areas in which the subjects were most experienced or interested on the make-up of the ontologies they produced.

7.4.3 PROTOCOL ANALYSIS¹⁷

7.4.3.1 Time per Page

A one-way within-groups ANOVA on the average time spent on each page over the four sessions was not significant ($F(3, 30) = 1.24$, $MS_{\text{error}} = 135.72$, $p > 0.05$). There was also no significant correlation between the amount of time spent, on average, on a page and the usability of the system reported.

There was a significant difference in the amount of time spent looking at different types of objects ($F(7, 2491) = 10.28$, $MS_{\text{error}} = 38640.52$, $p < 0.001$). Tukey-tests showed that subjects spent significantly less time on domains and individuals than on annotations, criteria or comparisons, and spend significantly less time on roles or classes than they did on annotations or comparisons.

Subjects also spent significantly longer on WebGrid pages than on those generated by APECKS directly ($t(2539) = -3.02$, $p < 0.001$). They spent significantly longer on pages with prompts (the 'Actions' page of roles and classes and the normal view of comparisons) than those without prompts ($t(594.48) = -5.66$, $p < 0.001$). They also spent significantly more time on pages on which they were filling in forms ($t(919.34) = -8.46$, $p < 0.001$).

¹⁷ The number of analyses made on the protocols collected and reported in this section increases the number of Type I (false positive) errors: no adjustments have been made to compensate for this.

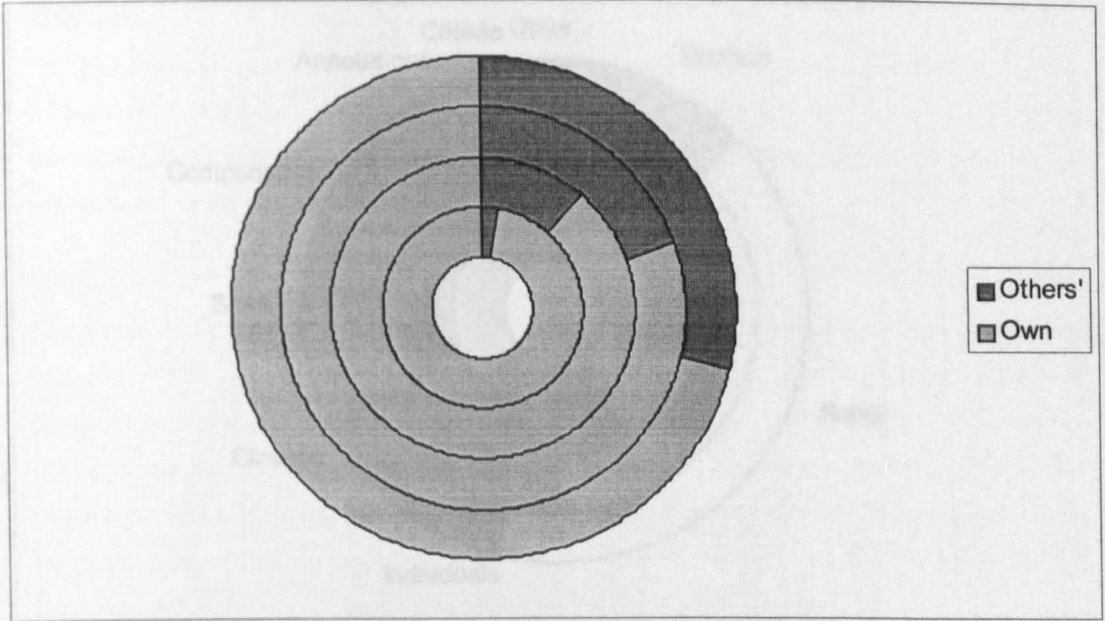


Figure 7.6: Changes in the proportion of visits made to others' roles over the sessions. The innermost ring is the first session.

7.4.3.2 Forms submitted

The proportion of page requests that were form submissions over the sessions also did not alter significantly ($F(3, 30) = 2.70$, $MS_{\text{error}} = 0.03$, $p > 0.05$), indicating that the subjects continued to enter information as much in the final as in the initial sessions. There was no correlation between the proportion of forms submitted and the reported usability of the system.

7.4.3.3 Ownership

The proportion of page requests that were visits to pages owned by others did change significantly over the sessions ($F(3, 30) = 9.34$, $MS_{\text{error}} = 0.02$, $p < 0.001$), with subjects spending a significantly larger proportion of their time looking at what other people had added in the final session (29% of the visits on average) than in the first (3% of the visits on average) ($t(10) = -4.17$, $p < 0.01$). The proportion of visits to others' information was significantly higher in the final session than in the second session (11% of the visits on average) ($t(11) = -4.05$, $p < 0.01$), indicating that the rise from the first to final session was not simply a matter of subjects receiving different instructions in the first session. These differences can be seen in the pie chart in Figure 7.6.

There was no correlation between the proportion of visits made to objects not owned by the subject and the reported usability of the system.

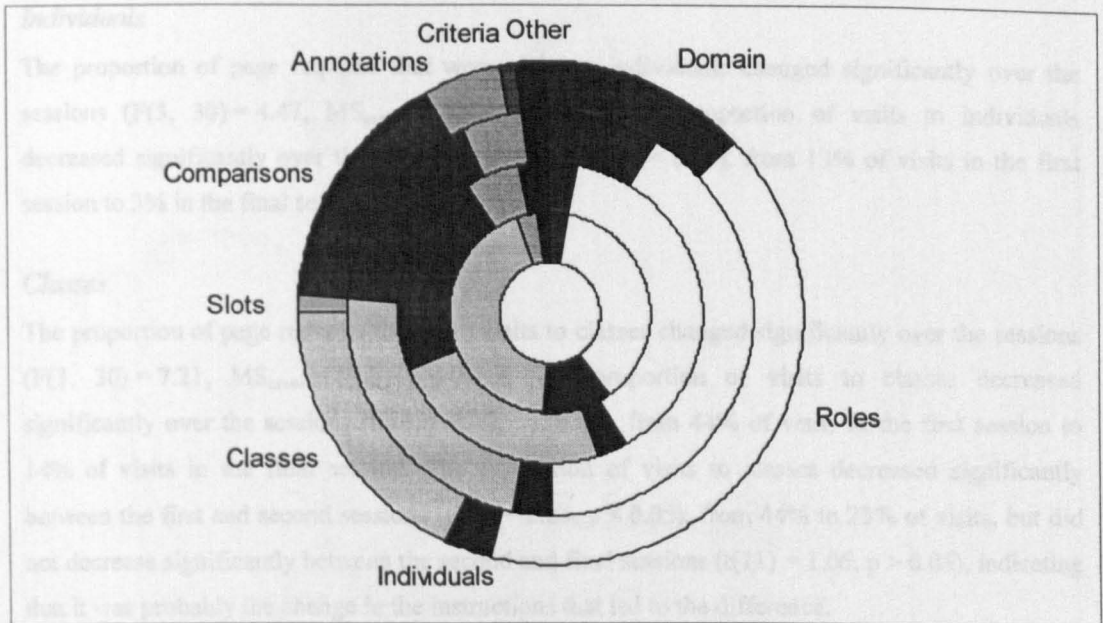


Figure 7.7: Changes in the proportion of visits made to different object types over the sessions. The innermost ring is the first session.

7.4.3.4 Object Type

The pattern of the types of objects visited varied over the sessions. There were no significant correlations between the proportion of visits to any type of object and the reported usability of the system. One-way within-subjects ANOVAs were carried out on the proportions of visits to each of the eight types of object (domain, role, individual, class, slot, comparison, annotation, criterion).

Domain

The proportion of page requests that were visits to the domain object changed significantly over the sessions ($F(3, 30) = 5.13$, $MS_{\text{error}} = 0.01$, $p < 0.01$). The proportion of visits to the domain increased from 4% to 13% of visits between the first and final sessions ($t(10) = -2.40$, $p < 0.05$). The proportion of visits to the domain increased from 3% to 13% of visits between the second and final sessions ($t(11) = -2.88$, $p < 0.05$), indicating that the rise from the first to final session was not simply a matter of subjects receiving different instructions in the first session.

Roles

The proportion of page requests that were visits to roles did not change significantly over the sessions ($F(3, 30) = 0.20$, $MS_{\text{error}} = 0.03$, $p > 0.05$).

Individuals

The proportion of page requests that were visits to individuals changed significantly over the sessions ($F(3, 30) = 4.47$, $MS_{\text{error}} = 0.01$, $p < 0.05$). The proportion of visits to individuals decreased significantly over the sessions ($t(10) = 2.39$, $p < 0.05$), from 13% of visits in the first session to 3% in the final session.

Classes

The proportion of page requests that were visits to classes changed significantly over the sessions ($F(3, 30) = 7.21$, $MS_{\text{error}} = 0.03$, $p < 0.01$). The proportion of visits to classes decreased significantly over the sessions ($t(10) = 4.70$, $p < 0.01$), from 44% of visits in the first session to 14% of visits in the final session. The proportion of visits to classes decreased significantly between the first and second sessions ($t(10) = 2.26$, $p < 0.05$), from 44% to 23% of visits, but did not decrease significantly between the second and final sessions ($t(11) = 1.06$, $p > 0.05$), indicating that it was probably the change in the instructions that led to the difference.

Slots

The proportion of page requests that were visits to slots did not change significantly over the sessions ($F(3, 30) = 1.82$, $MS_{\text{error}} = 0.00$, $p > 0.05$).

Comparisons

The proportion of page requests that were visits to comparisons changed significantly over the sessions ($F(3, 30) = 11.74$, $MS_{\text{error}} = 0.01$, $p < 0.001$). There was a significant increase in the proportion of visits to comparisons between the first and final sessions ($t(10) = -4.63$, $p < 0.01$), rising from less than 1% of visits to 17% of visits. The proportion of visits to comparisons increased significantly between the first and second sessions ($t(10) = -5.31$, $p < 0.001$), rising from less than 1% to 22%, but did not change significantly between the second and final sessions ($t(11) = 2.15$, $p > 0.05$), indicating that it was probably the change in the instructions that led to the difference.

Annotations & Criteria

The proportion of visits that were made to annotations did not change significantly over the sessions ($F(3, 30) = 1.30$, $MS_{\text{error}} = 0.00$, $p > 0.05$), and nor did the proportion of visits that were made to criteria ($F(3, 30) = 1.30$, $MS_{\text{error}} = 0.00$, $p > 0.05$).

7.4.3.5 WebGrid-II

A one-way within-groups ANOVA showed that there was no significant difference in the proportion of actions involving WebGrid across the sessions ($F(3, 30) = 1.44$, $MS_{\text{error}} = 0.03$, $p > 0.05$).

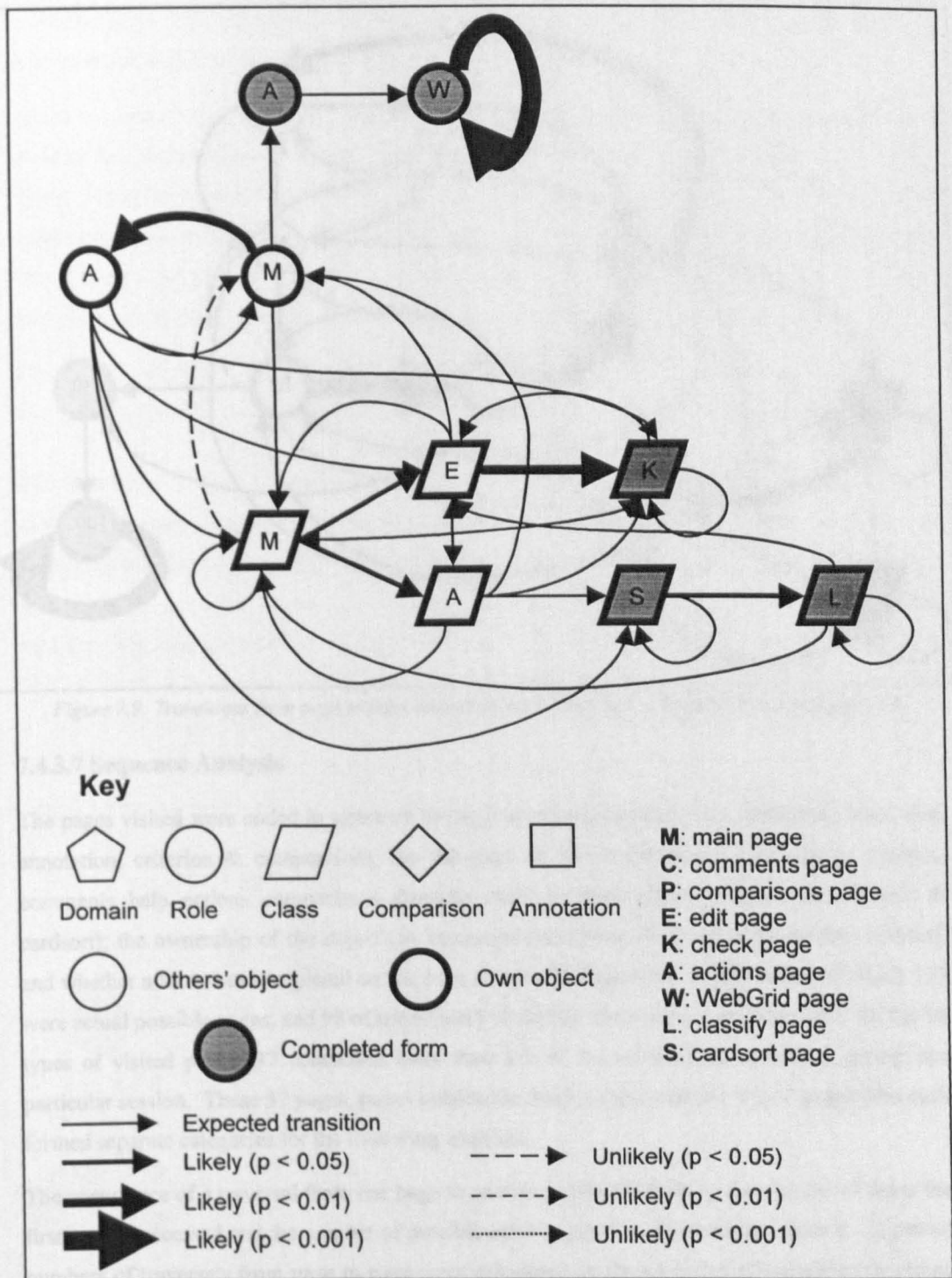


Figure 7.8: Transitions from page to page during the first session.

7.4.3.6 Prompts

A one-way within-groups ANOVA showed that subjects used pages with prompts for actions to a similar extent across the sessions ($F(3, 30) = 0.03$, $MS_{\text{error}} = 0.01$, $p > 0.05$).

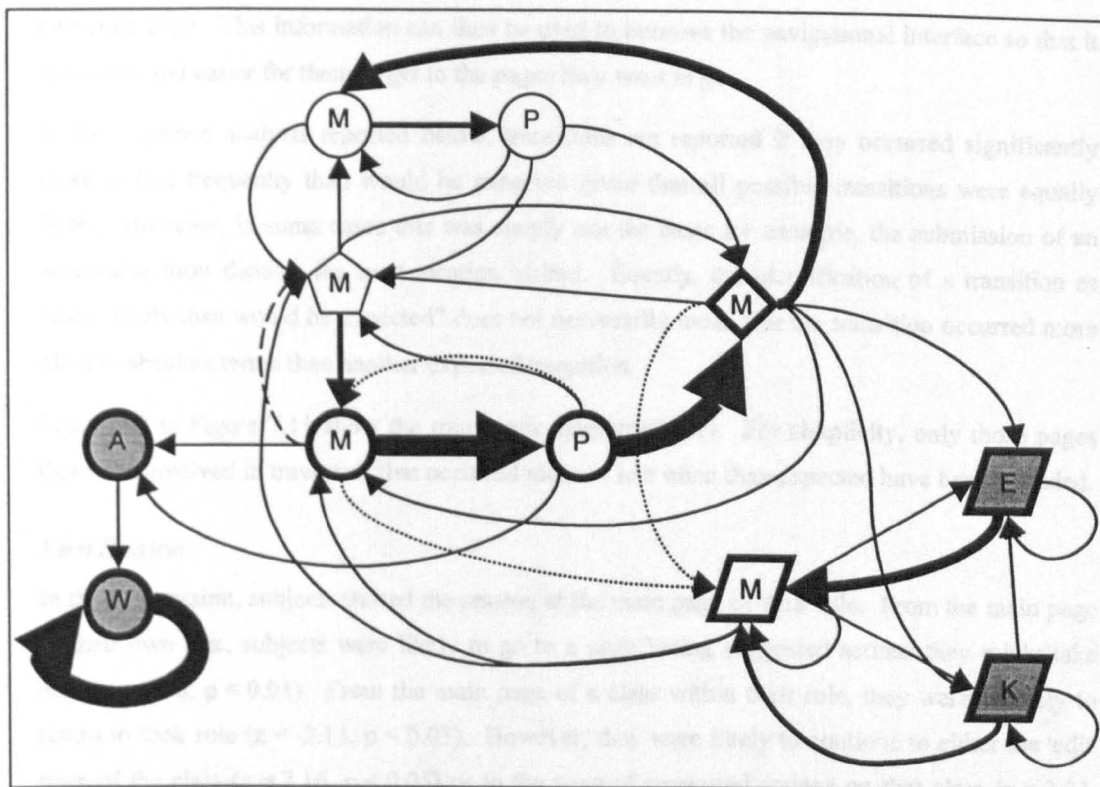


Figure 7.9: Transitions from page to page during the second session. The key is given in Figure 7.8.

7.4.3.7 Sequence Analysis

The pages visited were coded in terms of: the type of object (domain, role, individual, class, slot, annotation, criterion & comparison); the sub-page of the object (main, edit, check, changes, comments, help, actions, comparisons, domains, roles, gateway, compare, constraints, classify & cardsort); the ownership of the object (by the subject accessing the page or by another subject); and whether a form was completed on the page or not. This gave rise to 480 codes, of which 159 were actual possible pages, and 98 of which were visited by some subject at some point. Of the 98 types of visited pages, 37 comprised more than 1% of the visits either overall or during one particular session. These 37 pages, pages outside the main system, and the 'other' pages then each formed separate categories for the following analysis.

The occurrence of a traversal from one page to another is affected both by the number of times the first page is accessed and the number of possible other pages that can be visited from it. Expected numbers of traversals from page to page were calculated on the basis that all possible transitions (excluding those in which the subjects pressed the 'back' button of the web browser) were equally likely to take place, while those which were not possible did not occur at all. Residuals were calculated based on these expected values and standardised by session. This row-based sequence analysis helps to identify those paths regularly taken by users—where they want to go from a

particular page. This information can then be used to improve the navigational interface so that it is quicker and easier for them to get to the pages they want to go.

In the sequence analysis reported below, transitions are reported if they occurred significantly more or less frequently than would be expected given that all possible transitions were equally likely. However, in some cases this was simply not the case: for example, the submission of an acceptable form dictates the next location visited. Equally, the identification of a transition as “more likely than would be expected” does not necessarily mean that the transition occurred more often in absolute terms than another expected transition.

Figure 7.8 to Figure 7.11 show the transitions diagrammatically. For simplicity, only those pages that were involved in traversals that occurred more or less often than expected have been included.

First Session

In the first session, subjects started the session at the main page of their role. From the main page of their own role, subjects were likely to go to a page listing suggested actions they might take next ($z = 3.28, p < 0.01$). From the main page of a class within their role, they were unlikely to return to their role ($z = -2.13, p < 0.05$). However, they were likely to continue to either the ‘edit’ page of the class ($z = 2.16, p < 0.05$) or to the page of suggested actions on that class ($z = 2.31, p < 0.05$). From the ‘edit’ page, they were likely to continue to a page allowing them to add or remove annotations, criteria, instances or subclasses to the class ($z = 2.76, p < 0.05$).

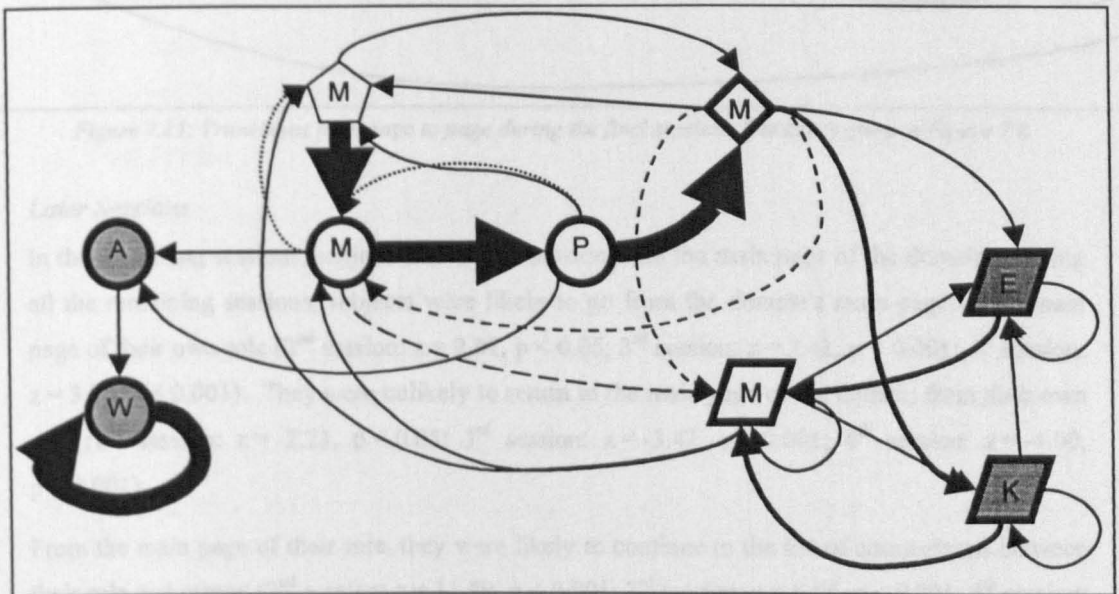


Figure 7.10: Transitions from page to page during the third session. The key is given in Figure 7.8.

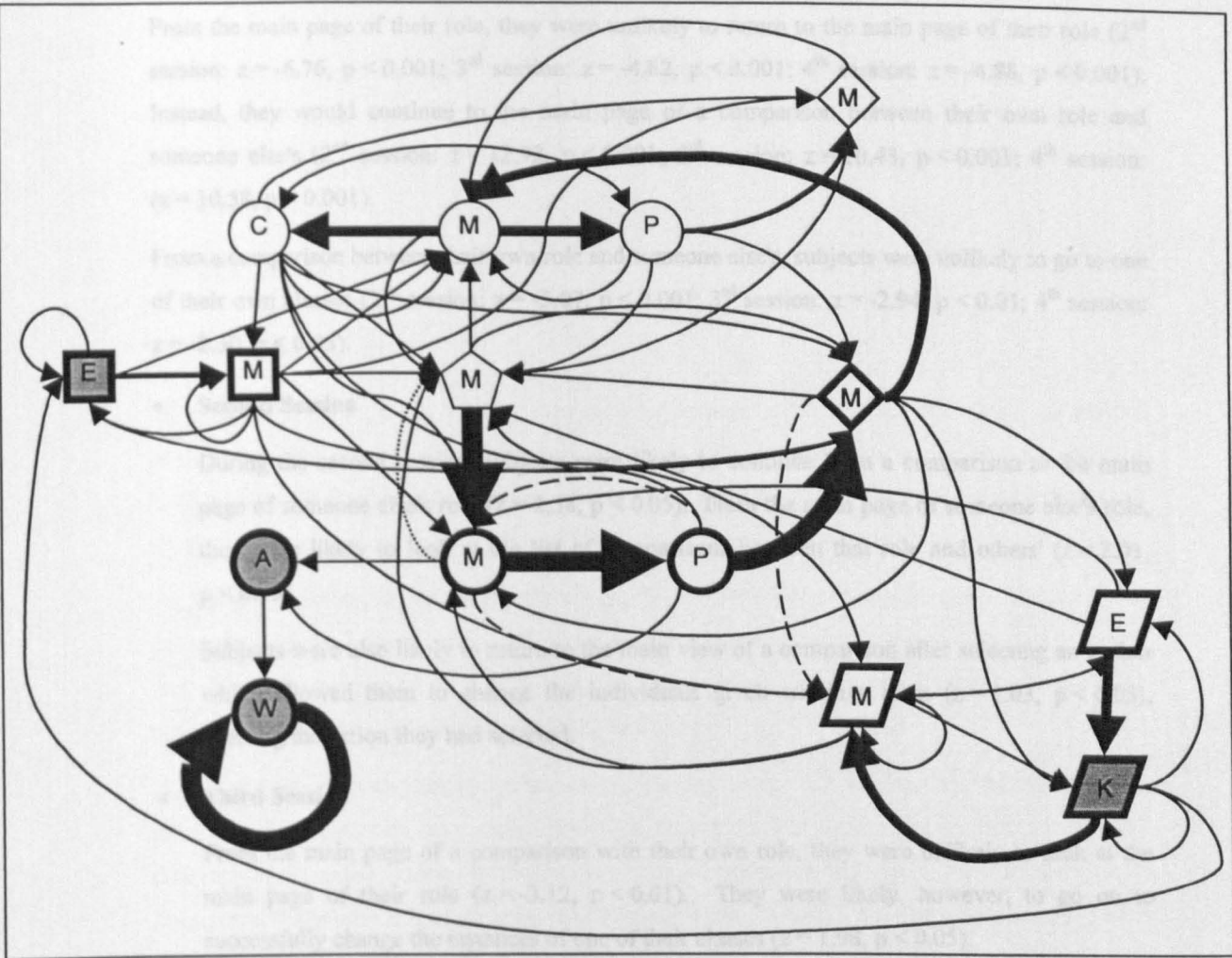


Figure 7.11: Transitions from page to page during the final session. The key is given in Figure 7.8.

Later Sessions

In the remaining sessions, subjects started the session from the main page of the domain. During all the remaining sessions, subjects were likely to go from the domain's main page to the main page of their own role (2nd session: $z = 2.01$, $p < 0.05$; 3rd session: $z = 3.41$, $p < 0.001$; 4th session: $z = 3.93$, $p < 0.001$). They were unlikely to return to the main page of the domain from their own role (2nd session: $z = -2.21$, $p < 0.05$; 3rd session: $z = -3.42$, $p < 0.001$; 4th session: $z = -4.00$, $p < 0.001$).

From the main page of their role, they were likely to continue to the list of comparisons between their role and others' (2nd session: $z = 11.89$, $p < 0.001$; 3rd session: $z = 8.06$, $p < 0.001$; 4th session: $z = 9.91$, $p < 0.001$). In both the second and final sessions, they were less likely than expected to proceed to one of their own classes (2nd session: $z = -3.37$, $p < 0.001$; 4th session: $z = -2.10$, $p < 0.05$).

From the main page of their role, they were unlikely to return to the main page of their role (2nd session: $z = -6.76$, $p < 0.001$; 3rd session: $z = -4.62$, $p < 0.001$; 4th session: $z = -4.88$, $p < 0.001$). Instead, they would continue to the main page of a comparison between their own role and someone else's (2nd session: $z = 12.92$, $p < 0.001$; 3rd session: $z = 10.43$, $p < 0.001$; 4th session: ($z = 10.58$, $p < 0.001$).

From a comparison between their own role and someone else's, subjects were unlikely to go to one of their own classes (2nd session: $z = -5.07$, $p < 0.001$; 3rd session: $z = -2.94$, $p < 0.01$; 4th session: $z = -2.30$, $p < 0.05$).

- **Second Session**

During the second session, subjects were likely to continue from a comparison to the main page of someone else's role ($z = 2.58$, $p < 0.05$). From the main page of someone else's role, they were likely to look at the list of comparisons between that role and others' ($z = 2.01$, $p < 0.05$).

Subjects were also likely to return to the main view of a comparison after selecting an option which allowed them to change the individuals given within a class ($z = 2.03$, $p < 0.05$), aborting the action they had selected.

- **Third Session**

From the main page of a comparison with their own role, they were unlikely to look at the main page of their role ($z = -3.12$, $p < 0.01$). They were likely, however, to go on to successfully change the instances of one of their classes ($z = 1.98$, $p < 0.05$).

- **Final Session**

From the comparison between their own role and someone else's they were unlikely to continue to one of their own classes ($z = -2.30$, $p < 0.05$), and likely to go to someone else's role's main page ($z = 2.83$, $p < 0.01$). From there, they were likely to proceed either to look at the list of comparisons with that role ($z = 2.60$, $p < 0.01$) or to the list of comments on the role ($z = 2.98$, $p < 0.01$). From the list of comparisons with another's role, they were likely to look at a comparison between that person's role and someone else's ($z = 2.10$, $p < 0.05$).

Other note-worthy transitions were from the 'edit' page of a class to a page allowing the subject to add or remove annotations, criteria, instances or subclasses to the class ($z = 3.08$, $p < 0.01$). Interestingly, most submissions resulting from such a page returned to the main page of the class ($z = 2.69$, $p < 0.01$), implying that the subjects were usually changing the instances of the class.

WebGrid-II

Through all the sessions, those who visited and submitted a WebGrid-II page were likely to continue to submit the pages (session 1: $z = 6.36$, $p < 0.001$; session 2: $z = 4.37$, $p < 0.001$; session 3: $z = 3.93$, $p < 0.001$; session 4: $z = 5.39$, $p < 0.001$). This led to extended sessions using WebGrid-II.

7.5 DISCUSSION

7.5.1 SYSTEM USABILITY

The analysis of the reported usability over the four sessions shows that subjects found the system easier to use over time. It seems that subjects found APECKS particularly easier to use after the third session, after they had been using it for around 1½ - 2 hours. The only activities that did not show evidence of getting easier over time were removing information and having discussions.

The reported usability of the system during the first session was not significantly correlated with the usability in the other sessions. This indicates that those who found the system hard to use at first did not necessarily find it hard to use later, and vice versa. This may be due to the differing tasks during the first session compared to the later ones.

The degree to which the subjects found APECKS easy to use did not generally depend on their previous experience with computers. However, during the first session, when the subjects were first introduced to the system, those who were more experienced with the WWW found it harder to use than those with less experience did. Perhaps either APECKS style of interaction, in which there are more forms than usual on normal WWW pages, or the relative lack of graphical content on the pages, confused those experienced with normal WWW applications. Those who were more experienced with email reported higher general usability in the second session, during which they were first introduced to discussion. Perhaps those used to email-style typed interaction found annotations easier to use. These effects of experience disappeared in later sessions.

The comments given by the subjects were very helpful in identifying areas in which APECKS could be improved. The problems the subjects encountered fell into three main areas:

- **Presentation:** The terminology used within APECKS led to particular problems, especially where the terminology was knowledge engineering-oriented. The layout and text used caused problems for some subjects. While they liked being able to focus in on a particular aspect of their role, they also suggested that an overview of the role should always be showing.
- **Navigation:** Subjects found it hard to navigate around the system, particularly without using the web browser's 'Back' button. The main page of a subject's role acted as a landmark for them and it was suggested that a link to this page should be available at all times. In addition, an 'Undo' button would have helped the subjects correct any errors they made. The time delay

between clicking on a link and the new page displaying was not a big problem, but this may have been because the APECKS server was close to the client computer in terms of network delay.

- **Discussion:** The subjects found it difficult to have discussions, often because of a reticence to comment on others' roles. They reported that would have preferred a more synchronous discussion mechanism.

7.5.2 ONTOLOGY QUALITIES

The roles had significantly more individuals, classes, hierarchies and annotations in the final session than they did in the first session. This shows that the subjects continued adding information steadily throughout the study: they did not stick with the individuals they had chosen originally, or with a single classification hierarchy. The subjects used the comparisons with other roles to select individuals and classes to add to their own roles. They also added annotations throughout the study. The number of class hierarchies within the roles increased over the sessions, while the number of partitions did not. This suggests that the subjects were creating classes as they came to mind, perhaps as a result of looking at comparisons with other people's roles, rather than only using the prompts to create subclass partitions as they did in the early sessions. The fact that extra slots and criteria were not created throughout the study suggests that the subjects did not understand how to use them, or found their use too complicated in the earlier sessions, and were put off by them.

The qualitative scales on which the seed and final subjects' roles were judged seemed to address the important qualities of ontologies and distinguished between the seed ontologies and those constructed by the subjects. Quantitatively, the 'better' roles (as judged qualitatively) tended to have both more individuals and more subclassifications of those individuals. The number of annotations made on a role seemed to decrease the knowledge engineer's perception of its qualitative goodness. Perhaps the fact that a knowledge representation is discussed undermines it as a useful ontology. Possibly, the annotations that were made were simply not useful. The relationship between annotations and decreased qualitative goodness may be an artefact of the fact that fewer annotations occurred on the (otherwise better) seed ontologies.

For the most part, the experimental design failed to generate biases in the ontologies produced by the subjects according to the tasks they were asked to carry out. The one exception to this was that those who had been told that the information would be used for a "Mammal-Spotter's Guide" addressed mammalian morphology more than those who were told it would be used as a dietary guide for zoos. It may be that the tasks were too closely related (they were all classification tasks) to give any difference, but this is interesting considering the amount of discussion that surrounds the task-specificity of ontologies (see Section 2.3).

7.5.3 PROTOCOL ANALYSIS

Unsurprisingly, subjects spent longer on pages when they filled in forms (WebGrid-II pages included) and on pages that contained prompts. On both these kinds of pages, there is a large amount of information to take in and a certain amount of consideration must be taken before proceeding to the next page. For these kinds of pages, it is true that 'think-time dominates load-time'. Types of objects which generate more information and more forms, such as annotations and comparisons, were thus visited for longer than those which do not, such as the domain and individuals and, to a lesser extent, roles and classes.

The average amount of time spent per page did not change over the sessions, and nor did the number of forms that were submitted. This indicates that subjects were still entering information at the same rate in the later sessions as in the earlier ones. Since the proportion of requests that were visits to annotations and criteria did not increase over the sessions, this suggests that the subjects were continuing to change their own roles over the sessions.

The subjects did display an increased amount of interest in other people's roles over the sessions. This was evident both in the increased proportion of visits to comparisons and to the domain, and in the increased proportion of visits to objects outside their own role.

The decrease in the proportion of visits to both classes and individuals indicates that subjects became less concerned with the details of their own (and others') role, and were more interested in objects that gave overviews of roles.

The sequence analysis shows that standard routes were taken through APECKS by the subjects. The distinction between the first session and the later sessions can be seen in the pattern of actions. During the first session subjects tended to visit the 'Actions' pages of their roles and classes. In later sessions, subjects followed the pattern of domain main page → own role's main page → list of comparisons with own role → comparison between own & others' role's main page. Both these patterns of transitions were those shown to the subject during the instructions. It is interesting that they stuck stubbornly with the latter pattern in the later sessions, despite the quicker possible jump straight from the domain's main page to a comparison's main page. This may have been because they wished to stick with a known sequence of actions, or because it was easier to identify comparisons with between their own role and other people's in this way.

7.6 CONCLUSIONS

Overall, the evaluation showed that, with a few provisos, APECKS was a usable system:

- The more the subjects used APECKS, the easier they found it to use.
- The subjects were able to produce knowledge representations using APECKS alone.

- The subjects used the knowledge acquisition support built into APECKS (including WebGrid-II) in order to produce their knowledge representations.
- From their comments, the subjects found the comparisons between their own and others' roles to be very useful, and used them in three main ways:
 - To improve their own role;
 - To investigate others' roles;
 - To make comments on others' roles.
- The subjects became less egocentric and more interested in other people's roles over the course of the experiment.

These findings are particularly positive considering the hurdles that the subjects had to overcome. They were given a relatively short amount of time to construct their ontologies, considering that APECKS is designed for extended use. Despite their biological background, the subjects were not specialists either on mammals or on taxonomic classification. They had no previous experience in knowledge engineering.

However, three aspects can be seen to require improvement:

- The presentation of information to users.
- The means of navigation through the system.
- The discussion system.

In addition, the generality of the conclusions drawn from this evaluation must be tempered with an understanding of the limitations that are placed on it. Firstly, without comparisons to other systems, it cannot be said that APECKS is better or worse than them. Secondly, the users, domain and usage timeframe were all limited within this study: use by knowledge engineers, different domains, and extended use may bring different problems.

The changes that should be made to APECKS in order to improve these areas are discussed in detail in Chapter 8.

CHAPTER 8 FUTURE WORK

8.1 SUMMARY

This thesis has described a methodology based on a constructivist view of knowledge engineering and a system that uses this methodology to support the collaborative construction, comparison and discussion of ontologies. In this final chapter, I indicate some of the areas into which future work could develop.

There are two main areas for future work based on this thesis. Firstly, APECKS itself as a system could be developed in terms of:

- Extension of the internal knowledge representation system used
- Integration with the text-based virtual environment and synchronous communication features provided by the MOO on which it is based
- Integration with other knowledge-based applications on the Internet, such as ontology servers, knowledge acquisition tools and ontology comparison tools
- Support for version control
- Extended evaluation of its use

Secondly, three research areas will inform the methodology and eventually the design of systems similar to APECKS. These are:

- Heuristics for ontology design
- Methodologies for comparing and combining ontologies
- Structures for discussions involving collaborative ontology construction

We are currently experiencing a period of rapid development and advance in computer-related knowledge and techniques. As with any computer-based research taking place now, the standards and technologies that are in use now are very different from those that were available at the beginning of the research. This chapter finally discusses the development of XML (eXtensible Markup Language) and associated standards, and their implications for both APECKS and for knowledge-based systems on the Internet in general.

8.2 INTRODUCTION

This thesis has presented both a theoretical discussion of collaborative ontological engineering and a practical demonstration of an application designed to support this process, APECKS. There are two areas into which this work can develop: the development and evaluation of methodologies for collaborative ontological engineering; and the evolution of APECKS itself.

In Section 8.3, I discuss four areas that are currently underdeveloped within APECKS. Firstly, the knowledge representation used internally within APECKS could be developed. Secondly, the knowledge structuring system could be integrated much more closely with the underlying collaborative virtual environment on which the system is based. Thirdly, the integration with other knowledge-based applications and resources on the Internet could be enhanced. Finally, there could be greater support for version control.

In Section 8.3.4, I discuss some of the theoretical areas in which development and evaluation is needed. Firstly, I discuss some of the design decisions that are made when developing ontologies and how these can be investigated further. Secondly, I describe methods for comparing and combining ontologies and how these may be developed. Finally, I highlight the need for formalisms for discussion during collaboration on ontology development.

This research involves technologies that have changed rapidly recently. Since the beginning of this research, computers have become more powerful; bandwidth and connectivity have increased; the World Wide Web has exploded in use and new technologies and standards have come into being. These have implications in the evolution of APECKS as a system, and also in the change in demands for applications. Section 8.5 discusses some of the relevant recent developments and their implications for this research.

8.3 EVOLUTION OF APECKS

APECKS as described within Chapter 5 and Chapter 6 is a working system that has been shown, from the evaluation given in Chapter 7, to support collaborative ontology construction by domain experts effectively. APECKS was developed more as a proof of concept than as a fully functional system. Therefore, there are areas of APECKS that have not been well developed and are discussed here.

- The internal knowledge representation (see Section 8.3.1)
- Integration with the virtual environment (see Section 8.3.2)
- Integration with other networked knowledge-based application (see Section 8.3.3)

- Support for versioning of ontologies (see Section 8.3.4)
- Extended evaluation (see Section 8.3.5)

8.3.1 KNOWLEDGE REPRESENTATION

The knowledge representation used within APECKS is based on the Frame Ontology used within Ontolingua (see Section 2.7 for a description) and is described in detail in Section 5.4.2. Five areas could be developed: classes and slots as frames; facets; explicit class hierarchies; axioms; and including ontologies.

8.3.1.1 Frames

Within the Frame Ontology (Karp & Gruber, 1997), individuals, classes, slots and facets are all frames: they can all have slots and belong to classes. This representation enables users to group classes and slots, as well as individuals, into classes, which gives meta-level knowledge representations. It also leads to the distinction on classes between 'instance slots', which are slots inherited by the individuals within the class, and 'own slots', which are those defined for the class itself.

Classes

In normal use in APECKS, it is possible to treat classes as frames: to classify them and assign slots to them. However, allowing users to do this was found to be confusing for the naïve domain experts during the pilot of the evaluation. For example, the distinction between the *types* of a class (the classes it belongs to) and the *superclasses* of a class is not readily apparent. Because of this, during the evaluation study described in Chapter 7, those features were disabled. The question then arises as to in what situations it is useful to be able to treat classes as frames.

One way in which classes are routinely grouped together is within subclass partitions. Within partitions, a group of subclasses is defined as classifying exclusively the instances of a superclass: no subclass within the partition can take an instance that is taken by any of the other subclasses within the partition. A partition can be seen as a class taking the subclasses as instances and slots defining whether the class is exclusive and/or exhaustive. In this case, then, the classification of classes constrains the instances of those classes.

Another example in which the grouping of classes can be seen is in the layout of schematic views (e.g. concept maps) of class hierarchies. The use of spatial organisation has been shown to be important in both representing and navigating knowledge (Marshall et al., 1991). One can imagine that this is particularly the case when there are several coexisting classification hierarchies and multiple inheritance. In this case, the classification of classes does not affect the

classes themselves in terms of the instances they can hold, but helps those using schematic views to gain an overview of the classification structure.

The classification and assignment of slots to classes can be seen to be useful, then, in some cases, but confusing in its current implementation within APECKS. In both the above examples, classes have either had individuals as instances or taken classes as instances. The latter can be termed meta-classes, as they act on a level above the normal classification level. The Frame Ontology allows for classes to have both individuals and classes as instances at the same time, and for slots to hold values for both individuals and classes at the same time. It is hard to conceive of a situation in which this would be useful. Making an explicit distinction between classes acting at the normal level and those acting at the meta-level may aid users in using both. Meta-level classes could be used to determine both the behaviour and the presentation of classes. Changing subclass partitions to work in this way would provide a more elegant internal representation and would facilitate both discussion about them and the representation of their change history.

Slots

The classification and assignment of slots to slots is not supported at all within APECKS. Slots instead have built-in properties that constrain the values that they can take: their domain (i.e. what classes define them), range, cardinality, inverse, symmetricalness, reflexivity and transitivity (see Section 5.4.4.3). These are the common slots of slots defined within the Frame Ontology. These slots of slots are distinct from facets in that they hold values that apply to the slot as a whole rather than for a specific class or individual.

The classification of slots can be useful in the same situations as for classes: both for the presentational grouping of slots and to affect their behaviour by constraining the values they can take. This facility should be incorporated into APECKS.

8.3.1.2 Facets

Facets give values on slots for particular frames. Within APECKS, facets are not represented at all. Instead, slots have built-in properties that constrain the values that can be taken by specific classes and individuals: range, cardinality, 'same' slots and 'must have' values (see Section 5.4.4.3). These common facets are defined within the Frame Ontology. For advanced ontological engineering, it is likely that knowledge engineers require more functionality, including user-defined facets.

8.3.1.3 Hierarchies

The hierarchies within APECKS are currently automatically generated on the basis of the class membership of individuals. A subclass is defined as a class that holds a subset of the instances of its superclass. Where there are not enough individuals defined, this leads to unexpected and

inaccurate hierarchies. This can lead to insights into the structure of the domain and thus to the elicitation of further, contrary, examples in cases where the classification is being elicited afresh. However, it can be frustrating in cases where the classification hierarchy is already known.

APECKS should be extended to allow the explicit specification of subclass-superclass relationships. In some cases, it is not possible or desirable to specify the superclasses of a class, so these should only be set explicitly by the user rather than fixed by the system. Consistency checking, as described in Section 5.4.4.3, could indicate where the user-defined class hierarchy was not consistent with the classification of the individuals.

One of the reasons this facility was not used in APECKS originally was the difficulty in handling the addition of instances to classes. There are two alternative ways of dealing with adding instances if the superclasses of a class are fixed:

- Only allow the addition of individuals that are instances of the superclass
- Allow the addition of any individual, and add that individual to the superclass as well

Both mechanisms are useful at different times. When an individual is being newly classified, it is simplest to let the class add the individual to all its superclasses. Where the classification structure is being reorganised, it is perhaps better to constrain the addition of instances. To handle this, the distinction between those individuals that do belong to a superclass and those that do not should be indicated when the user is selecting instances for the class. Users could also state that the selected individuals should be added as instances of the superclasses of the class, as well as to the class itself.

8.3.1.4 Axioms

Axioms are rules that can constrain or compute class instances or slot values. APECKS does not have any facility for specifying axioms. All constraints (see Section 5.4.4.3) are built-in rather than user-defined, which limits the extensibility of the representation. For sophisticated use by knowledge engineers, it is necessary to allow them to specify axioms. This involves the interpretation of a knowledge representation language, such as KIF, in which axioms can be expressed. For domain experts in particular, there is also a requirement for support in creating axioms using the language. While it is unlikely that domain experts will be able to write axioms directly without guidance and a good interface, it is quite likely that they would be able to articulate rules they wish to encode.

Axioms are also useful for testing for membership of infinite classes, such as 'integers'. Such classes are primarily used to specify the range of slots. In APECKS, primitive values are used instead of these classes. However, in some cases, it is useful to constrain ranges in unexpected ways, such as only allowing a slot to take even integers. In these cases, infinite classes are useful.

8.3.1.5 Including Ontologies

At the moment, within APECKS, ontologies within a domain are separate from ontologies within other domains. There is no means of importing ontologies or copying them between domains. This means that users cannot take advantage of the ontologies they, or others, have created in other domains, a facility that is common in other ontology servers (see Section 2.6.1.1). For example, an ontology describing the knowledge acquisition community draws on ontologies describing people, papers, organisations and so on (Benjamins & Fensel, 1998). It is helpful to reference these ontologies rather than reinvent or painstakingly copy them.

The complication in APECKS centres on the definition of multiple roles within any particular domain. Importing a particular role into an ontology constrains the ontology to an extent that might not be acceptable. An alternative would be to reference equivalent classes within the roles in a certain domain. For example, if a user is trying to constrain the range of a slot 'supervisors' within an ontology to only researchers, they could specify the range as "the class 'researcher' in the role 'University staff', or equivalent class". By comparing the roles within the 'people' domain, consensual or corresponding classes from other roles, which also indicate whether a person is a researcher, could be used interchangeably. The instances that were allowed as values would not change according to the role used, by the definition of *consensual* and *corresponding* classes (see Section 5.4.5), but the ontology used to describe people might.

8.3.2 MOO INTEGRATION

The APECKS system is built on top of a publicly available MOO database, the LambdaMOO core database (see Section 3.4). Thus, many features from MOOs are available naturally within APECKS. However, at the moment no advantage is taken of them: the MOO simply provides network connections, persistent user identities and a rapid prototyping environment. There are two areas in which further advantage can be taken of MOOs: using the virtual environment and integrating synchronous communication.

8.3.2.1 Virtual Environment

The APECKS system presents users with a frame-based view of a structured, semi-formal knowledge representation. Knowledge engineers may be used to this type of view of knowledge, from other ontology servers such as Ontolingua (see Section 2.6.1.1). From the evaluation study detailed in Chapter 7, it is clear that while some domain experts found the semi-formal presentation familiar and easy to understand, others found it confusing.

Users of ontology servers work with knowledge on two levels. Firstly, there is direct access to knowledge content, involving discovering and discussing information about the individuals within the domain. Secondly, there is meta-level access to knowledge representation, involving

maintaining and discussing how the knowledge content is structured. The emphasis on each of these levels depends on how much domain knowledge is included in the knowledge structure (see Section 2.3). Within ontology servers (see Section 2.6.1.1), the emphasis is on structure over domain knowledge whereas in corporate memory systems (see Section 2.6.1.2), the emphasis is on the recording of domain knowledge over structure. Similarly, knowledge engineers are more interested in the structure of a domain than its content, while the opposite is true of domain experts.

These two levels require different kinds of presentation of the same internal knowledge representation. When users are interested in knowledge content, they should be able to access this information directly, without concerning themselves with what is represented as a slot or class. On the other hand, when they are interested in manipulating the representation, they need access to these higher level structures. This is particularly true of domain experts: knowledge engineers may be able to digest semi-formal representations of knowledge, but domain experts are not used to working in this way.

As we have seen in Section 4.4.2, spatial representations of hypertext sites can help users navigate through the information by taking advantage of their ability to make cognitive maps of real spaces. MOOs are collaborative virtual environments (see Section 3.3.1). They provide an environment with which people can interact: they can move between rooms, pick up objects and so on. It may be that utilising the virtual environment to give a spatial representation of the knowledge would facilitate navigation through the system.

There are two ways in which spatial representations can help, operating at the two levels described above: structure and content.

- **Structure**

Spatial representations of the structure of knowledge have to support the construction and navigation of classes and slots. At this level, abstract representations of structures can be used that are independent of the domain being represented. For example, an untangled classification hierarchy might be represented as a tree, which users can climb to move to subclasses and on which leaves represent individuals. Slots might be represented as pieces of string linking individuals together. Knowledge acquisition tasks can also move closer to their real world application: carrying out a card sort (see Section 2.5.1.2) can involve putting virtual cards into groups.

Text-based virtual environments are useful for this type of abstract representation of complex structures because they do not have to adhere to real world spatial rules (see Section 3.3.1). For example, tangled class hierarchies (and indeed hypertext structures), due to their complexity and interconnectedness, often do not naturally form three dimensional or

Euclidean structures. Representing these in a graphical environment would be difficult, but text-based virtual environments can represent these easily. The issue then is whether such complex and unfamiliar representations are useful for navigation by users, or if they are too distant from reality to enable them to construct cognitive maps of the space.

- **Content**

Spatial representations of knowledge content have to support the navigation of information within a domain. At this level, concrete representations of elements within the domain can be used: the nature of the representation is dependent on the domain. For example, a knowledge representation about cars would generate a virtual environment populated with cars, and looking at one of them would reveal information about it. The organisation of the environment might reflect an aspect of the knowledge structure, such as by organising the cars into showrooms dependent on their manufacturer, but the entirety of the structure would not be apparent. While such a representation chiefly facilitates information retrieval, it could also be used to construct knowledge representations, such as by driving a car into a different showroom (changing the make), respraying (changing the colour) or swapping the radio (changing the sound system).

The representation of roles within such a spatial metaphor is also interesting. The automated detection of equivalence of classes and slots across roles would allow representations of individuals to work across roles to a certain extent. Where roles focus on different, but coexistent, aspects of individuals, the representations might be merged without problems. For example, information about the engines of the different cars might come from a different role to that about buyer options within them: both could be represented within the single representation or virtual showrooms. However, in some cases, different types of representations may have to be built for different roles within a domain, as well as for different domains. Each virtual representation involves a handcrafted translation between the internal knowledge representation and the virtual environment: it may be that the navigational support arising from such representations is not worth the effort.

There are three issues arising from using these two types of representations, structure-oriented and content-oriented. Firstly, the type of representation used should depend on the user's preferences. This would allow users to focus on the aspect of knowledge in which they are most interested. Secondly, there needs to be a way to navigate between the two types of representation or mix them if appropriate. Different tasks will be easier in different representations, so swapping between them would be helpful. Finally, users should be able to invent new metaphors easily. They should be able to define translations between internal knowledge representations and spatial representations so that they can use metaphors they feel comfortable with: skyscrapers instead of trees; a suburban street instead of showrooms.

As with many of the possibilities for future work described in this chapter, any development in this direction needs to be tempered by evaluations of the utility of the feature. Some research questions in this area are:

- Do spatial representations actually help navigation through information structures?
- Is the ease of navigation within a spatial representation dependent on the similarity of the representation to the real world?
- What types of abstract spatial representations are best for representing knowledge structures?

8.3.2.2 Synchronous Communication

MOOs provide synchronous text-based communication between users. Synchronous communication allows users to converse more naturally and dynamically than asynchronous annotations. The enhancement of the basic MOO database included the provision of WWW-based synchronous communication (see Section 5.4.1). However, communication conducted in this way was neither easily available through navigation nor integrated into the knowledge structuring facilities within APECKS during the evaluation described in Chapter 7.

The virtual environment described in Section 8.3.2.1 above provides a context in which such communication could take place. For example, conversations within a room representing a particular class, individual or slot could be automatically archived as an annotation on that frame, if those involved in the discussion wish. A particular issue arising from this feature is that, according to the methodology on which APECKS is based, discussion should mainly arise because of comparisons between roles. The virtual environment described above therefore needs to provide contexts in which two roles can be compared side-by-side within the environment.

Much of the discussion on synchronous communication within this thesis has focused on text-based communication due to the ease with which it can be archived and indexed. With advances in voice-recognition software and the increase in communication bandwidths, it is now feasible that more expressive audio communication could take place without losing the advantages of text-based communication. Such communication should be integrated into the virtual environment and archived for easy access.

8.3.3 NETWORK INTEGRATION

APECKS can act as both an HTTP server, to provide information to users, and an HTTP client, to gather information from other resources (see Section 5.4.1). Currently, the only resource accessed in this way is WebGrid-II, which is used over the Internet both as a knowledge acquisition tool (see Section 5.4.4.2) and for the comparison of roles (see Section 5.4.5). Three types of systems could be accessed by APECKS over the Internet:

- **Ontology servers** could provide alternative roles within domains. By gathering ontologies within a particular domain from several ontology servers, it would be possible to compare ontologies across systems.
- **Knowledge acquisition tools** could facilitate the construction of ontologies by domain experts, in the same way as WebGrid-II.
- **Ontology comparison tools** could be used to provide different types of comparisons between roles, based on which APECKS could prompt both changes to them and discussion about them. WebGrid-II is already used in this way.

One important issue arising from such integration is how knowledge-based systems provide information to other networked applications. With WebGrid-II, the results of knowledge acquisition were available as hidden fields within the web page provided to the user in a structure unique to WebGrid. The results of comparisons were only available as images, which could not be analysed by APECKS although they could be displayed directly to the user. The Knowledge Interchange Format (KIF: Genesereth & Fikes, 1992) was designed precisely to aid in the exchange of knowledge in bulk between systems. KIF would therefore be appropriate for use in retrieving large portions of knowledge, such as entire ontologies from ontology servers. The Knowledge Query and Manipulation Language (KQML: Finin, Labrou & Mayfield, 1997) can be used to exchange smaller amounts of information between agents. Using this would allow incremental and interactive exchanges between applications, for example to keep distributed representations consistent during knowledge acquisition, and for quick queries to be made of ontologies held within ontology servers.

Taking the use of these applications to their logical extreme, it may be that eventually tools like APECKS will not need to support the storage, construction or comparison of ontologies internally at all. Instead, such collaboration tools will manage interactions between ontology servers, KA tools and comparison tools, for example by supporting the construction of an ontology using a knowledge acquisition tool, and then storing the ontology into an ontology server. The roles of such a system would be both coordination between distributed tools and the provision of support for discussions between users.

8.3.4 VERSIONING SUPPORT

APECKS records every change that occurs to objects represented within it (see Section 5.4.6.2). This is currently used to provide design rationales by attaching annotations to changes. However, it could also be used to provide versioning facilities and the ability to return roles to previous states.

At the moment, APECKS is passive in supplying information about changes that have occurred since the user's last visit. Changes are only shown to users if they specifically look at them, although the changes that are shown by default are only those that have occurred since the user's last visit to the object. Instead, it may be useful to have a subscription mechanism whereby users can subscribe to areas in which they are interested and are informed on connection of any changes that have happened in those areas.

8.3.5 EXTENDED EVALUATION

The evaluation of APECKS described in Chapter 7 had several limitations, each of which should be addressed by future evaluation studies. Example areas for future evaluations are:

- Other ontology servers: is APECKS an improvement over existing ontology servers?
- Other ontology servers used in conjunction with a collaborative virtual environment: does APECKS' integrated discussion system provide more than similar functionality from existing systems?
- APECKS without comparison support: do automated comparisons help users to discuss and develop ontologies?
- Use by knowledge engineers: do knowledge engineers have the same experience of APECKS as domain experts?
- Different domains and tasks: does the domain or task used change the way APECKS is used?
- Extended timeframe: how does APECKS perform with extended use?

Each of these evaluations could use the same evaluation techniques as those used in the evaluation described in Chapter 7, and could thus be compared with the study discussed here.

8.4 THEORETICAL RESEARCH

As indicated in Section 7.2, APECKS is an implementation of a methodology. As such, not only can the system be developed further, but theoretical research also needs to be carried out that can inform its development. The methodology described in this thesis is based on a constructivist view of ontological engineering described in Section 2.4. Within this theory, collaborative ontological engineering can be seen as a design process in which decisions are made by those constructing ontologies. The various design options should be assessed against evaluative criteria. In this way, knowledge engineers can select ontologies based on their requirements in a particular context.

Three theoretical areas require further research:

- How ontologies are actually designed (Section 8.4.1)
- Methodologies for comparing and combining ontologies (Section 8.4.2)
- Structures for discussions about ontology design (Section 8.4.3)

8.4.1 ONTOLOGY DESIGN

The first area that needs development is to develop a number of evaluative criteria against which ontologies can be judged. In order to explore the types of decisions that are made by ontological engineers, it is necessary to look at the methodologies they use in developing and evaluating ontologies. Such a study has been carried out by Jones, Bench-Capon & Visser (1998). They identified some distinctions between and evaluations of ontologies that could be converted into criteria, such as:

- The task type, problem-solving method and domain-type of the ontology

*From CommonKADS and KACTUS (Schreiber, Wielinga & Jansweijer, 1995;
Wielinga et al., 1994)*

- Coverage: is every concept of interest covered?
- Granularity: is every relevant distinction made?

From Plinius (Mars et al., 1994)

- If something has a fixed position in time and/or space, it's an instance: if not, it's a concept

From Mikrokosmos (Mahesh, 1996; Mahesh & Nirenburg, 1995)

- Similarity: a subclass must be of the same type as its parent
- Specificity: a subclass must have some difference that distinguishes it from its parent (the *differentiæ*)
- Opposition: the subclasses of a concept are incompatible with each other
- Unique semantic axis: the subclasses of a concept can be constrained to differ from the parent in some common property or 'axis'

From MENELAS (Bouaud et al., 1994; Bouaud et al., 1995)

- Substantial sortals (which are both countable and temporally stable) should be defined as classes while non-substantial sortals should be defined as slots

From Guarino, Carrara & Giaretta (1994)

More detailed analysis of ontological engineering methodologies needs to be carried out in order to identify to what extent different ontological engineers subscribe to different heuristics when developing ontologies. These heuristics could be made explicit within APECKS and users encouraged to indicate those to which their ontologies comply. The comparison and discussion of different ontologies within the same domain, as supported by APECKS, also uncovers the heuristics used by ontological engineers.

8.4.2 COMPARING AND COMBINING ONTOLOGIES

Within the methodology described in this thesis, the exploration of the design space of possible ontologies within a domain is carried out by constructing separate ontologies, comparing them and discussing them. As an ontology server, APECKS should make ontologies available to knowledge engineers, but knowledge engineers usually simply require a single ontology within a domain. A single ontology could be generated in one of two ways:

- The knowledge engineer could select one of the roles on the basis of the criteria the role fulfils
- An ontology could be automatically generated on the basis of several roles

Multiple ontologies from different domains are often combined to give an ontology for a specific context. This is done by identifying high-level concepts that appear across ontologies, and using these to link them together. Combining multiple ontologies from the same domain cannot use this simple approach as several of the ontologies may represent the same distinctions in different ways, leading to ontologies that are internally inconsistent.

The first problem in combining ontologies is to identify the differences and similarities between them. Methods for comparing ontologies based on the Shaw & Gaines (1989) classification of relationships between experts are used within APECKS (see Section 5.4.5). These methods highlight terminological and conceptual similarities between ontologies. Further research on this area might develop other methodologies for comparing ontologies that took into account the structure of the classification hierarchy or used the properties of slots to identify similarities. In addition, the pair-wise comparison of ontologies leads to scalability problems as the number of comparisons combinatorially explodes. Further research should investigate ways of comparing multiple ontologies at the same time, with systems only generating them on demand.

Once the comparison of roles has been carried out, the conglomerate ontology can be generated. Lin (1994) suggests an algorithm for merging knowledge in information systems that could be used in this context. The algorithm weights statements about knowledge according to the number of representations that agree with the statement. Majority support for a statement means that it is included within the merged representation. The algorithm proposed necessitates the representation

of knowledge in the same way (using the same terms) across the representations that are merged. However, future research might extend this algorithm to give a methodology for combining ontologies that use different terms. The following issues are worth noting:

- Roles within a domain could be given different weights for different merges. Conglomerate ontologies might thus be created based on a single role, with extra information added from others, or formed by combining many roles on an equal basis.
- The first step in the creation of a conglomerate ontology in which all contributing roles are given the same weight might involve the representation of consensual classes and slots from across all the roles. However, often consensual frames are scattered within the knowledge representation rather than forming even a skeletal classification hierarchy. Even if frames which correspond to each other across roles (i.e. are the same concept given different terminology) are added, the conglomerate ontology would still probably not be sufficient.
- Adding all contrasting frames (i.e. those which do not have equivalents in any other roles) might make the conglomerate ontology unnecessarily large and confusing.
- In the addition of corresponding and conflicting frames (i.e. those which match on either terminology or underlying concept, but not both), decisions have to be made as to which role gives the desired representation. These decisions could be automated using the weights attached to the different roles, but, if they are equal or near equal, the user will have to make a decision.
- The comparison and combination of multiple roles is substantially harder than simply comparing and combining two ontologies. A frame that corresponds to a frame in another role might be in conflict with a frame in a third while in contrast to frames in another. Managing these multiple comparisons is an area for further research.

8.4.3 DISCUSSION FORMALISMS

One of the areas that was indicated as needing improvement by the evaluation of APECKS was support for discussion (see Chapter 7). As noted in Section 5.3.3.4, no specific formalism was imposed on APECKS, although a distinction was made between free annotations and evaluative criteria. The reason for the lack of specific formalism was, as discussed in Section 3.6, the minimal correspondence between the various formalisms used in the design rationale literature.

Structuring discussion based on a formalism aids both the retrieval of information from archives and the logic of the discussion itself, in theory. The question is whether these assertions are true in practice and in what contexts these advantages are apparent. Further research might study the discussion of knowledge engineers and domain experts in the construction of ontologies in order

```
<rock name="Granite">
  <grainsize value="large"/>
</rock>
```

Figure 8.1: Example XML syntax.

to discover the Speech Acts that are used. Such an analysis could then lead on to the development of a formalism for discussion in the collaborative ontological engineering process.

8.5 RECENT DEVELOPMENTS

Increasingly, hypertext documents are generated automatically from information within a database. APECKS is one example; search engines another common example. Within these data-based systems, the production of hypertext documents is composed of three interlinked stages.

1. **Generation** of external formal or semi-formal representations of information
2. **Translation** of the semi-formal representations into human-readable form
3. **Structuring** of the human-readable information into documents

During each of these stages, the user's identity and preferences can be taken into account in order to change the information finally available within the document and its presentation and navigation. The user's identity may change how they view the domain generally, as we have seen within APECKS. During translation, the user's identity may determine the nature of the human-readable form (graph, table, diagram or text). During structuring, the user's identity may determine the order of presentation of information.

In this section, I discuss recent developments in standards and technologies and their impact on how data-based systems like APECKS generate hypertext. These are:

- XML (Section 8.5.1)
- Schemas (Section 8.5.2)
- Styling Languages (Section 8.5.3)
- Linking Languages (Section 8.5.4)

8.5.1 XML

The past year has seen the emergence of XML (Extensible Markup Language, World Wide Web Consortium, 1998a) as a standard for transmitting rich structured information over the Internet. XML is a simplified subset of SGML (Standard Generalised Markup Language) designed for use on the WWW. In HTML, the elements that can be used, such as H1 (heading level one) or P (paragraph) are fixed within a specification. SGML and XML are, however, *meta-Markup Languages* that allow users to specify their own elements according to their needs.

The basis of an XML document is a number of hierarchically structured elements, each with a number of attributes. For example, in Figure 8.1 both `rock` and `grainsize` are elements, while `name` and `value` are attributes. The element `rock` contains the element `grainsize` and takes the value "Granite" for the `name` attribute, while the element `grainsize` is an empty element, taking the value "large" for the `value` attribute.

The advantage that XML brings over HTML is the ability to use structures relevant to the domain in order to describe the content of a page. In HTML, information about a rock could be buried within a paragraph, the name given in a heading and properties given in lists. In XML, it can be explicitly indicated using a `rock` element. This makes it easier to identify the meaning of text within a document for search engines (see Section 4.5.1.2) and web-based knowledge acquisition agents (see Section 2.5.2.3).

The obvious next step for APECKS in terms of serving information over the WWW is to produce XML rather than HTML. XML representations can be much closer to the internal representations used by APECKS than the HTML representations currently produced. This has advantages in the ease, speed and flexibility of the generation of information. Generating XML also means that it will become easier for other applications to retrieve information from APECKS. However, before APECKS and other knowledge-based applications on the Internet start to use XML, the knowledge engineering community needs to agree on the elements and attributes that can be used to represent knowledge. KIF and KQML (see Section 8.3.3) might be used as bases for knowledge representation markup languages that could be used to exchange knowledge between applications and agents.

8.5.2 SCHEMAS

The elements and attributes that are used within an XML document depend on the information that is encoded within it. Information providers may invent elements and attributes as they find the need for them. However, if the same elements and attributes are used by those talking about the same things, applications can be pre-programmed to process the information held. This has two large applications:

- **Presentation of information**

For example, a graphics package might understand elements representing lines and points. If their names were changed, the information within the XML document could not be displayed in the same way.

```

<!ELEMENT rock (grainsize?)>
<!--ATTLIST rock
      name      ID                      #REQUIRED>

<!ELEMENT grainsize EMPTY>
<!--ATTLIST grainsize
      value      (small|medium|large)    #REQUIRED>

```

Figure 8.2: Example XML DTD for Figure 8.1.

- **Information retrieval**

Specialised information brokers focusing on a single domain will search pages for a particular element. Again, if a different element name were used, the information encoded within the page could not be retrieved.

For this reason, *schemas* are used to give explicit definitions of the elements that are used within a particular domain.

XML 1.0 defines a basic way of defining the elements that can be used within an XML document, the elements they can contain, the attributes they take and the values those attributes take. This set of definitions is known as the DTD (Document Type Definition). A possible DTD for Figure 8.1 is shown in Figure 8.2. The DTD shown defines that a `rock` element can have a `grainsize` element inside it and has to give a value for the attribute `name`, which serves as a unique identifier within the document for the element. `Grainsize` elements, on the other hand, cannot contain any other elements and must give a value for the attribute `value`, which can take one of the values "small", "medium" or "large".

As can be seen in Figure 8.2, DTDs use a different syntax from the body of XML documents. Schemas use the same nested element structure as XML to provide definitions. There are already several schemas proposed for giving meta-information about XML document content: XSchema (St. Laurent, 1998), XML-Data, DCD (Document Content Description: Bray, Frankston & Malhotra, 1998). In the main, these schemas have been designed to mirror the basic document type description (DTD) defined within XML 1.0.

These schemas offer the following extra features above DTDs:

- **Documentation**

Schemas allow for the documentation of element types, attribute types, content models, attribute values and so on. These enable applications to access human-readable descriptions of a particular Markup Language syntax.

- **Attribute Value Constraints**

Within basic XML 1.0 DTDs, attributes can either be unconstrained, allowed to take only particular enumerated values, or constrained to take certain roles (such as identifiers or entity references). Schemas often give the opportunity to constrain attribute values by their type (e.g. integers, colour specifications, URLs). They can also sometimes define further constraints on attribute values depending on the type, such as the range of integers that can be taken; the brightness of the colour; or the location of the URL.

- **Inheritance**

Some schemas attempt to reuse element and attribute specifications through inheritance. For example, in XSchema, attributes defined outside element definitions apply to all elements defined within that section. Similarly, in DCD, element definitions can inherit content models and attributes from other elements.

There is a great deal of similarity between schemas and ontologies. Where schemas define elements and attributes, ontologies define classes and slots. As with ontologies, schema authors for SGML are traditionally specialists, but with XML, schema authoring may become a mainstream activity, just as domain experts are beginning to create ontologies directly. Finally, the construction of both schemas and ontologies is a design activity that will need to be supported by tools for collaboration.

The knowledge engineering community has a lot of experience in the design and maintenance of ontologies. This experience can be transferred to the design and maintenance of schemas. XML languages that mirror knowledge representation languages can be used as schema languages, to define elements and attributes in XML languages for knowledge content. Schema languages based on knowledge representation languages offer additional benefits over the schema languages currently proposed. Most significantly, they give a user-extensible means of defining axioms for element and attribute content. These can be used during validation of XML documents to constrain element and attribute content, but, more importantly, the definition of axioms enables inferences to be made over XML content. This means that the knowledge available on the Internet need not be limited to that explicitly stated within XML documents.

The collaborative design of schemas raises the same issues as the collaborative design of ontologies. As demonstrated in Section 6.5, one of the ways in which APECKS can be used is in the collaborative creation of schemas for XML. However, the use of XML can also benefit from the association of XML languages to each other using systems like APECKS. The simplest example of such use is the support of translations of XML element and attribute names for different languages. More complex associations might indicate similarities between schemas

```

<p>
/name/ is a /colour/ /intrusive-extrusive/ rock
#if (#intrusions in /types/)
, found /depth/.
#else
.
#endif
It has a /grainsize/ grainsize due to its /speed-of-cooling/ cooling.
</p>
<p>
/name/ contains the following minerals:
/minerals/
</p>

```

Figure 8.3: A template for a view of a rock.

designed for different tasks or by different communities, and allow translations between them according to the preferences of the user.

8.5.3 STYLING LANGUAGES

The generation of human-readable material from semi-formal data like that provided in XML involves the application of heuristics over the data. For example, the presentation of a reference at the end of a paper varies according to whether it is a journal paper, conference paper, book chapter or technical report.

The presentation of information based on simple script-like style sheets is becoming more common within HTML pages. HTML has moved away from the use of presentational elements such as `I` (italics) and `FONT` (specifying the font of the element's contents) towards structural elements such as `EM` (emphasis) and `P` (paragraphs). Two stylesheet languages are currently proposed for use with XML:

- **Cascading Style Sheets (CSS)**

CSS (World Wide Web Consortium, 1996; World Wide Web Consortium, 1998b) can determine the appearance of elements according to their unique identifiers and their situation relative to other elements. Through CSS, an author or user can define traditional typesetting attributes of the text, such as the font, colours, borders, padding, alignment and so on. CSS is therefore very useful for presenting textual information.

- **Extensible Stylesheet Language (XSL)**

XSL (World Wide Web Consortium, 1998c), is based on DSSSL (Document Style Semantics and Specification Language: ISO/IEC, 1996) but specified in a declarative style using XML. As well as the ability to format text precisely in the same way as CSS, it can be used to restructure documents, changing the order in which information is presented.

```
<p>
Granite is a whitish intrusive rock, found deep down. It has a coarse grainsize
due to its slow cooling.
</p>
<p>
Granite contains the following minerals:
<ul>
  <li>plagioclase feldspar</li>
  <li>quartz</li>
  <li>muscovite</li>
</ul>
</p>
```

Figure 8.4: HTML generated for the rock Granite from the template specified in Figure 8.3.

Within an earlier version of APECKS, a similar approach to XSL was taken in the generation of human-readable versions of data from internal knowledge representations. A simple scripting language was used in order to generate HTML fragments from the slots of individuals. Within this scripting language, slot values could be simply inserted, or small MOO code programs could be used to generate suitable text on their basis.

Figure 8.3 shows a template given in this scripting language for viewing a rock. Given the rock Granite, this would generate the HTML shown in Figure 8.4.

With APECKS producing XML instead of HTML, as suggested in Section 8.5.1, it is possible to move the majority of this processing to the client-side, and instead supply default external style sheets to the users. This has the advantage that user's own style sheets can be used to give the layout they want for any sections, and that standard tools for constructing stylesheets can be used.

The next step for the translation of data into more user-friendly presentations is to be able to associate small programs, such as Java applets, with XML elements. These programs will be able to process the information within elements in a more sophisticated manner than stylesheets and produce more flexible presentations, such as graphs, or even 3D models.

8.5.4 LINKING LANGUAGES

The process of structuring hypertext documents involves bringing together information from several sources either before or after the application of styles to the information. This involves identifying the data that is required by the user.

An earlier version of APECKS supported the automated creation of hypertext documents in four steps:

1. Data was selected in terms of specific individuals (e.g. the rock Granite), instances of a class (e.g. all coarse-grained rocks) or the value of a slot (e.g. the rocks which hold quartz).


```
Laddered Grid I talks about the following rocks:
/list #rock.instances using Rock information with name/
```

Figure 8.5: A template for a document listing the rocks within Laddered Grid I

2. A view was selected to format each type of data. Each view was specified using the scripting language mentioned in Section 8.5.3 above.
3. The structure was specified, again using a simple scripting language that could test for the presence of data and the user's identity.
4. Links within the document, and between the document and other resources were specified externally and in a general form.

Figure 8.5 shows a template for a document listing the rocks in Laddered Grid I. The template specifies that the document should consist of some text (Laddered Grid I talks...) followed by a list of all the instances of the class `rock` given in the format specified by `Rock information`, a view which is specified by the template given in Figure 8.3. It was also possible to specify sections, subsections and to insert other existing documents into a document.

Figure 8.6 shows the definition of links from the document. It specifies that the text "Laddered Grid I" should be linked to the URL given; that the names of all instances of the class `rock` should be linked to the normal view of themselves; and that the names of all instances of the class `mineral` should be linked to the `Mineral information` view of themselves.

The preceding examples show a means by which authors could specify hypertext documents without knowing all the information that will go into the final document. This effectively separates the provision of information from the publishing of information and allows different users to view the same information in different formats, without repeating the information.

There are two developments that may be of use to the construction of documents based on distributed XML information resources.

- **External Entities**

XML documents can define external resources that are brought in as part of the current document. These are termed entities and are intended to produce modular documents by

```
"Laddered Grid I" link to
  http://www.psychology.nottingham.ac.uk/research/ai/sisyphus/
  Protocols/lg/laddered_grid_1.html
#rock.instances link to themselves
#mineral.instances link to Mineral information
```

Figure 8.6: The link specification for the document given in Figure 8.5.

```

<!DOCTYPE doc [
  <!ENTITY header SYSTEM "header.ent">
  <!ENTITY footer SYSTEM "footer.ent">
]>
<doc>
  &header;
  ...
  &footer;
</doc>

```

Figure 8.7: Example entity use within an XML document.

combining existing snippets of XML in the same way that *server-side includes* and sometimes HTML FRAMES (and the more recent HTML OBJECTS) are commonly used. Figure 8.7 shows how a predefined header and footer can be inserted into a document.

External entities offer two advantages over server-side includes. Firstly, the resources that make up the entities can be located anywhere, not necessarily on the host machine. This makes it easy to draw resources in from information providers around the globe. Secondly, entities are handled by the client, which reduces the load on the server. However, this does increase the load on the client and network traffic as more resources have to be requested to create the document.

- **XLink**

XLink (World Wide Web Consortium, 1998d) defines a number of attributes that can be used to define links between documents or sections of documents. Resources, for the purpose of XLink, can be whole documents, or parts of documents specified by XPointer (World Wide Web Consortium, 1998e).

With XLink, links can be defined as either *inline* or *out-of-line*. Inline links, such as the `A` element in HTML, indicate that the current resource is one end of the link. Out-of-line links, on the other hand, are defined in a separate document from any of the linked resources. For example, a single document within a WWW site could define all the links between pages on the site. A further classification of links is made within XLink in that *extended* links allow the linking of several resources together, as different options for the user to choose from or to indicate an interlinked group of resources.

XLink also defines a few behaviour semantics. Authors can specify whether links are automatically followed or have to be actuated by the user. They can also specify how the new resource is displayed: embedded within the resource, replacing the resource or within a new context. Links that are automatically followed and in which the remove resource is embedded clearly perform the same function as external entities. However, they are more flexible than entities since XPointers can be used to retrieve only portions of documents.

Entities and XLink can be used to combine information from several distinct resources into a single document without first retrieving those documents wholesale. As with XSL stylesheets, they can be generated using standard XML tools due to the use of XML syntax.

Entities and embedded XLinks simply create larger XML documents that can be formatted in the same way as smaller elements, using XSL. There is currently no way to specify the behaviour of other types of link, and it is left up to the client as to how this is done. To a certain extent, this depends on how the rest of the document is rendered: if as a normal document, links may appear as buttons or pull-down menus; if as a 3D environment, they may be exits to other rooms.

For knowledge-based applications on the WWW, entities and XLink allow information to be truly distributed across systems. By simply including a link in an XML representation of an ontology, it will be possible to automatically include ontologies or portions of ontologies stored on other ontology servers.

8.6 CONCLUSIONS

This chapter has reviewed some of the areas of future work for both APECKS and the methodology on which it is based. APECKS could be developed in several ways:

- The knowledge representation could be expanded, involving:
 - the treatment of classes and slots as frames, allowing them to be classified and have values attached to them;
 - the addition of facets to store values on slots for certain frames;
 - the explicit specification of classification hierarchies (i.e. superclass-subclass relations);
 - the specification of axioms to constrain class membership and slot values; and
 - the inclusion of ontologies from other domains within roles.
- The knowledge management aspects of APECKS could be more highly integrated with the MOO on which APECKS is based. There are two aspects to this. Firstly, APECKS could use the text-based virtual environment available within the MOO to give a spatial representation of the knowledge it stores. Secondly, users could be encouraged to communicate synchronously as well as asynchronously, and this communication archived as annotations.
- APECKS could be more highly integrated with other knowledge-based systems on the Internet. In particular, APECKS could take better advantage of ontology servers as a source of roles; knowledge acquisition tools to help the construction of ontologies; and ontology comparison tools to compare roles.

- APECKS could use the rationale creation system currently in place to provide facilities for retrieving old versions of roles and to allow users to be actively informed of changes that occur to areas they are interested in.
- Future evaluations of APECKS could be used to inform its further development.

In addition, there are three areas in which further research is needed in order to expand the methodology expounded in this thesis:

- Research into the heuristics based on which ontological engineers design ontologies, the similarities and differences of these heuristics and their validity and utility for final ontologies.
- Research into methodologies for comparing and combining ontologies so that conglomerate ontologies can be produced semi-automatically.
- Research to identify the Speech Acts that are used by collaborating knowledge engineers so that a design rationale formalism for collaborative ontological engineering can be developed.

Finally, this chapter has discussed some recent developments that impact on the way APECKS and similar systems work, and the function they could play in the future. One of the major developments is simply the ability to have information presented to users in a manner that suits them. However, it also seems likely that the widespread adoption of XML will open new opportunities for knowledge-related activities in two main areas: the development of schemas and incidental knowledge acquisition and information retrieval from the Internet.

In traditional hypertext authoring, the authors know what information is available and in what way it can be viewed because they are the only sources of the information and the only sources of the views. With distributed information, authors may construct hypertext documents from a dispersed set of information. This is particularly the case for 'automated authors', such as search engines or information brokers, which face three problems:

- **Locating relevant resources**

The first step for an automated author is to locate relevant resources holding information of interest to the user. As a basis, most information brokers, such as Ontobroker or classified directories like those discussed in Section 4.5.1.1, allow information providers to register their information (which may be vetted for relevance) centrally. In automated search engines such as those discussed in Section 4.5.1.2, this is supplemented by periodic surveys of the WWW, following links from one page to another.

RDF (Resource Description Framework: World Wide Web Consortium, 1998f) seeks to supplement these methods by providing rich and relevant metadata giving further information

about the content of the page. In addition, the use of a particular DTD or schema indicates the subject matter of the page. Both these advances mean that automated search engines should become better at finding relevant material on the WWW.

- **Extracting the information**

The second step for an automated author is to extract information from the resource. It is difficult to extract information from HTML pages as the element structure generally indicates layout rather than the content of the page. There are two exceptions to this.

- Pages which are generated on the basis of some underlying data structure often produce regular HTML which an application can be programmed to parse and interpret.
- Pages can have hidden information attached to them, in the form of hidden form fields as in WebGrid, HTML comments, or in extra attributes or elements that are ignored by normal web browsers, such as are used in Ontobroker.

The development of XML facilitates the extraction of information from resources by automated authors. XML information is more likely (although not guaranteed) to use element structures that are relevant to the information contained within the page. The relevant elements can be located and extracted for use in the final document produced by the automated author.

XML information may be associated with schemas that define axioms. These axioms can be used to infer information not explicitly stated within the resources. This technique is used within Ontobroker (Fensel et al., 1998). Multiple XML languages may be defined within the same domain, giving multiple, associated schemas. Making the associations between these schemas explicit, as with applications like APECKS, also allows automated authors to expand the set of information available to them without forcing information providers to make everything explicit within a single resource.

- **Constructing documents**

Finally, automated authors must put together a coherent summary of the information that has been found according to the user's preferences. Previous efforts at automated authoring have necessarily focused on the above problems, resulting in documents that often do not actually contain the information required by the user, only pointers to resources which may. The increased ease of locating and extracting information enables information brokers to begin to pay attention to the presentation of information to users.

As the kind of intelligent, ontology-based information brokering described here grows, the processes that APECKS supports currently can be divided into five distinct areas:

1. Storing and serving schemas
2. Storing and serving knowledge content (information about individuals)
3. Supporting construction and editing of schemas
4. Supporting acquisition of knowledge content (information about individuals)
5. Supporting comparison and discussion of schemas

These five processes should be supported in the future by three types of application.

- **WWW servers** store and serve schemas as well as knowledge content within domains. They need to support the retrieval of sub-resources, portions of information within a single XML document, but can rely on XML browsers to dynamically generate human-readable views of the information.
- **XML browsers** support the navigation of XML documents while **XML editors** support their editing. Some browsers and editors may be specialised to certain XML dialects: for example, there may be specialised editors that only support the creation of documents about molecules. General XML browsers and editors need to use DTDs or schemas in order to display information in domain-specific ways and to guide the authoring of valid XML documents. Knowledge acquisition applications should produce both schemas and knowledge content in XML format for storage by WWW servers.
- **Collaboration supporters** aid the maintenance, comparison and discussion of distributed information and information structures. They need to be able to pull information in from a variety of sources, relate the different schemas to each other and support discussion about them. Not only does this support the creation and maintenance of schemas, but also the intelligent location and extraction of information through associating related schemas with each other.

APECKS already supports each of these aspects to a certain extent. However, it is in the latter role, as a supporter of collaboration between users, that its strength lies, and that will be focused on during its future development.

8.7 FINAL WORD

The problem of how to help people to share knowledge with each other is going to become more pressing over the next few years. People do not simply need more *information*: the growth of information technologies, especially the Internet, over the last decades has, some would say,

provided too much already. Knowledge is information that is delivered in the right form and at the right time so that people can use it to take decisions: the key to delivering information in the right way is to give it structure so that its purpose and relevance can be assessed.

Structuring information involves identifying the important elements within the domain: this can be facilitated by knowledge-acquisition techniques originally intended for the construction of knowledge-based systems. However, the essential difference between information and knowledge is that it is adaptive: it is in a form that is most applicable to the person accessing it for the task they are trying to achieve. As people have different biases, opinions, perspectives and so on, so the way they think about a domain differs. Capturing those differences not only enables us to present people with knowledge in a way that suits them, but it can also be used to help people to understand how other people think about a domain, and alleviate misunderstandings between them.

The system that is the focus of this thesis, APECKS, has a number of unique features that allow it to support the representation of differing views on a domain, the identification of where the differences lie between them, and the discussion of those differences.

- **Constructivist perspective**

The design of APECKS has been informed by a constructivist perspective that recognises that there are many ways of viewing the same domain. This contrasts with the classical perspective taken within most similar systems that assumes that there is only one correct way to represent knowledge within a domain.

Existing ontology servers hold unrelated ontologies that can only be accessed for changes by those to whom permission has explicitly been given. On top of this, preference is given to the existing state of affairs, with changes having to satisfy certain constraints, such as consistency with the rest of the representation or the agreement of all the other collaborators. This may lead to stagnant and restricted ontologies.

APECKS, on the other hand, is based on personal ontologies, representing an individual's understanding of a domain. Creating such ontologies is quick and easy, with support for copying and adapting the whole or part of other peoples', and anyone can create them. The ontologies can exist in a state of inconsistency within themselves, or with other individual's ontologies, so that the reasons for these inconsistencies can be explored. The emphasis in the creation of ontologies within APECKS is on change and exploration: a brainstorming process. This can lead to dynamic and diverse ontologies.

- **Direct knowledge-acquisition support**

Existing ontology servers give support to knowledge engineers in the construction of ontologies. APECKS is geared towards the support of domain experts, who are not necessarily experts in the construction of ontologies. APECKS currently carries out basic knowledge acquisition itself. Other, more sophisticated, techniques can be slotted into APECKS as they become available in internet-accessible tools. WebGrid (Gaines & Shaw, 1996a & 1996b), for example, is already used to give support for repertory grids within APECKS.

- **Automated ontology comparison**

The real innovation of APECKS, however, lies in its support for collaboration between users through the comparison of their personal ontologies. APECKS automatically compares the personal ontologies constructed by its users in terms of the consensus/conflict/correspondence/contrast classification put forward by Shaw & Gaines (1989). Ontologies are not just compared in terms of whether they are consistent with each other or not, but the degree of consistency with each other. In addition, it is not simply the way in which individual concepts are represented that is considered, but also the larger structure in which they sit.

Users of APECKS are prompted to act or communicate with each other based on the comparisons made between their personal ontologies. If correspondence (different terminology being used for the same concept) is detected, for example, the users who constructed the ontologies would be prompted to either change the term to bring them into line with the other or to start a discussion about why different terms were being used.

Discussion based on comparisons between ontologies has two useful consequences. Firstly, domain-related detail is uncovered to give richer, more detailed ontologies. Secondly, criteria used in the construction of ontologies are made explicit. This explicit statement of the criteria under which the ontologies are constructed provides meta-information about their purpose that is usually left implicit.

- **Use of networked resources**

While current ontology servers can be accessed by users either through a hypertext interface or programmatically, their focus is, as their name suggests, on serving information: they do not access other networked resources. APECKS, on the other hand, is envisioned as acting as a client as well as a server, accessing network-accessible knowledge acquisition applications and other ontology servers. Currently, it uses WebGrid (Gaines & Shaw, 1996a & 1996b) to support knowledge acquisition. In the future, it is envisioned that the ontologies held by other ontology servers would be accessed by APECKS and used just like the other personal ontologies APECKS holds. In this way, APECKS positions itself in the centre of the network of knowledge on the Internet.

- **Evaluation**

The final novel aspect of APECKS is that it is an evaluated system whose use, under experimental conditions, has been monitored and analysed. This has demonstrated both that there are a number of techniques for evaluating novel systems and that such evaluation can yield interesting and useful results.

This chapter has indicated a number of ways in which APECKS could be further developed and the challenges that lie with changing demands in the future. However, even in its current state, APECKS offers innovative and powerful support to the collaborative-design task faced by ontological engineers and in the future it will prove a strong basis for support for living ontologies.

REFERENCES

- Austin, L.C., Liker, J.K. & McLeod, P.L. (1993) Who Controls the Technology in Group Support Systems? Determinants and Consequences. *Human-Computer Interaction* 8, 217-236.....79
- Bateman, J.A., Magnine, B. & Rinaldi, F. (1994) The Generalized Italian, German, English Upper Model. Presented at *The Eleventh European Conference on Artificial Intelligence (ECAI'94) Workshop on Comparison of Implemented Ontologies*, Amsterdam, The Netherlands, 8-12 August 1994.....7
- Bell, G. (1995) Knowledge modelling and Adaptive hypertext. In *Proceedings of the 1995 Postgraduate Conference*, Psychology Department, University of Nottingham. URL http://www.psyc.nott.ac.uk/~grb/papers/pgconf-paper_ToC.html.....67
- Bellotti, V. (1993) Integrating Theoreticians' and Practitioners' Perspectives with Design Rationale. In *Proceedings of the 1993 conference on Human Factors in Computing Systems (INTERACT'93 and CHI'93)*, Amsterdam, The Netherlands, 24-29 April 1992.....59
- Bellotti, V., MacClean, A. & Moran, T. (1991) *Generating Good Design Questions*. Technical Report EPC-91-136, Rank Xerox Research Group, Cambridge, UK.58
- Benford, S., Greenhalgh, C. & Lloyd, D. (1997) Crowded Collaborative Virtual Environments. In *Proceedings of ACM Computer-Human Interaction'97 (CHI'97)*.....39
- Benford, S., Snowdon, D., Brown, C., Reynard, G. & Ingram, R. (1997) Visualising and populating the Web: Collaborative virtual environments for browsing, searching and inhabiting Webspace. *Computer Networks and ISDN Systems* 29(15), 1751-1761..... 71, 72, 73, 78
- Benford, S., Snowdon, D., Greenhalgh, C., Ingram, R., Knox, I. & Brown, C. (1995) VR-VIBE: A Virtual Environment for Co-operative Information Retrieval. In *Proceedings of Eurographics'95*, 30th August - 1st September, Maastricht, The Netherlands. 349-360..... 73, 78
- Benjamins, V.R. & Fensel, D. (1998) Community is Knowledge! in (KA)². In B.R. Gaines & M. Musen (eds.) *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'98)*, Banff, Canada, April 18-23, 1998. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW98/benjamins/>..... 20, 153, 192
- Bentley, R., Appelt, W., Busbach, U., Hinrichs, E., Kerr, D., Silkel, K., Trevor, J. & Woetzel, G. (1997) Basic support for cooperative work on the World Wide Web. *International Journal of Human-Computer Studies* 46, 827-846..... 68, 72
- Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H.F. & Secret, A. (1994) The World Wide Web. *Communications of the ACM* 37(8), 76-82.....68
- Bersot, O., El Guedj, P-O., Godéreaux, C. & Nagues, P. (1996) A Conversational Agent to Help Navigation and Collaboration in Virtual Worlds. In *Proceedings of Collaborative Virtual Environments '96 (CVE'96)*, Nottingham, UK, 19-20th September 1996.....39
- Bieber, M., Vitali, F., Ashman, H., Balasubramanian, V. & Oinas-Kukkonen, H. (1997) Fourth generation hypermedia: some missing links for the World Wide Web. *International Journal of Human-Computer Studies* 47, 31-65.....65
- Bouaud, J., Bachimont, B., Charlet, J. & Zweigenbaum, P. (1994) Acquisition and Structuring of an Ontology within Conceptual Graphs. In *Proceedings of ICCS'94 Workshop on Knowledge Acquisition using Conceptual Graph Theory*, University of Maryland, College Park, MD. 198
- Bouaud, J., Bachimont, B., Charlet, J. & Zweigenbaum, P. (1995) Methodological Principles for Structuring an Ontology. *IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, 19-20 August, 1995..... 198

- Boy, G. (1996) The Group Elicitation Method: An Introduction. In N. Shadbolt, K. O'Hara & G. Schreiber (eds.) *Advances in Knowledge Acquisition: Proceedings of the 9th European Knowledge Acquisition Workshop (EKAW'96)*, Nottingham, UK, May 1996. Springer-Verlag: Berlin.15
- Bray, T., Frankston, C. & Malhotra, A. (1998) *Document Content Description for XML*. W3C Note. [WWW Document] URL <http://www.w3c.org/TR/NOTE-dcd>..... 203
- Brown, P. (1988) Linking and searching within hypertext. *Electronic Publishing* 1(1), 45-53.74
- Bruckman, A., Heiner, J., Hudson, G., Samnick, D., Skwersky, A. & Tamm, S. (1996) *MacMOOSE* [WWW document]. URL <http://asb.www.media.mit.edu/people/asb/MacMOOSE/>.....49
- Buckingham Shum, S. & Hammond, N. (1994) Argumentation-Based Design Rationale: What Use at What Cost? *International Journal of Human-Computer Studies* 40(4), 603-652.50
- Buckingham Shum, S. (1996) Design Argumentation as Design Rationale. *The Encyclopedia of Computer Science and Technology*, vol. 35, supp. 20, 95-128. Marcel Dekker Inc: NY. 50, 58
- Buckingham Shum, S. (1997) Balancing Formality with Informality: User-Centered Requirements for Knowledge Management Technologies. *AAAI Sprint Symposium on Artificial Intelligence in Knowledge Management*, Stanford University, Palo Alto, California, March 24-26 1997. AAAI Press. 29, 66
- Buckingham Shum, S. (1997) Negotiating the Construction and Reconstruction of Organisational Memories. *Journal of Universal Computer Science: Special Issue on IT for Knowledge Management* 3(8), 899-928. URL <http://kmi.open.ac.uk/~simonb/sbs-jucs97.pdf>.....35
- Buckingham Shum, S.J., MacLean, A., Bellotti, V.M.E. & Hammond, N.V. (1997) *Graphical Argumentation and Design Cognition*. Technical Report KMI-TR-25, Knowledge Media Institute, The Open University. Also in *Human-Computer Interaction*. URL <http://kmi.open.ac.uk/kmi-abstracts/kmi-tr-25-abstract.html>.....55
- Carlstrom, E.-L. (1992) *Better Living Through Language: The Communicative Implications of a Text-Only Virtual Environment, or, Welcome to LambdaMOO* [FTP document]. URL <ftp://ftp.lambda.moo.mud.org/pub/MOO/papers/communicative.txt>.....43
- Chandrasekaran, B., Josephson, J.R. & Benjamins, V.R. (1998) The Ontology of Tasks and Methods. In B.R. Gaines & M. Musen (eds.) *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'98)*, Banff, Canada, April 18-23, 1998. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW98/chandra/>8
- Chen, L.L.-J. & Gaines, B.R. (1996) Knowledge Acquisition Processes in Internet Communities. In *Proceedings of the 10th Knowledge Acquisition Workshop (KAW'96)*. Banff, Canada, Nov. 9-14, 1996. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW96/chen/ka-chen-gaines.html>.....92
- Chen, L.L.-J. (1996) Chronological Awareness Tools: CHRONO and Meta-CHRONO. In *Proceedings of the 10th Knowledge Acquisition Workshop (KAW'96)*. Banff, Canada, Nov. 9-14, 1996. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW96/chen/kawchrono.html>92
- Cherny, L. (1995) The Modal Complexity of Speech Events in a Social MUD. *Electronic Journal of Communication* 5(4). URL <ftp://bhasha.stanford.edu/pub/cherny/ejc.txt> 43, 44
- Chung, P.W.H. & Goodwin, R. (1994) Representing Design History. In J.S. Gero & F. Sudweeks (eds.) *Artificial Intelligence in Design'94*, 735-752. Kluwer Academic Publishers..... 52, 54
- Cockburn, A. & Jones, S. (1996) Which way now? Analysing and easing inadequacies in WWW navigation. *International Journal of Human-Computer Studies* 45, 105-129. 71, 72

- Conklin, E.J. & Yakemovic, K.C.B. (1991) A Process-Oriented Approach to Design Rationale. *Human-Computer Interaction* 6, 357-391. 52, 54
- Conklin, J. & Begeman, M.L. (1988) gIBIS: A Hypertext Tool for Exploratory Policy Discussion. *ACM Transactions on Office Information Systems* 6(4), 303-331. Also in *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'88)*. ACM Press. 52, 53
- Conklin, J. (1987) Hypertext: An Introduction and Survey, *IEEE Computer*. 66
- Crow, L.R. & Shadbolt, N.R. (1998) IMPS – Internet Agents for knowledge engineering. In B.R. Gaines & M. Musen (eds.) *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'98)*, Banff, Canada, April 18-23, 1998. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW98/crow/> 20
- Curtis, P. & Nichols, D.A. (1993) MUDs Grow Up: Social Virtual Reality in the Real World. In *Proceedings of the Third International Conference on Cyberspace*. Austin, Texas. URL <ftp://ftp.lamba.moo.mud.org/pub/MOO/papers/MUDsGrowUp.ps> 42
- Curtis, P. (1992) Mudding: Social Phenomena in Text-based Virtual Realities. In *Proceedings of the 1992 Conference on the Directions and Implications of Advanced Computing*, Berkeley. Also available as Xerox PARC technical report CSL-92-4. URL <ftp://ftp.lambda.moo.mud.org/pub/MOO/papers/DIAC92.ps> 40, 45, 93, 94
- Curtis, P. (1996) *LambdaMOO Programmer's Manual For LambdaMOO Version 1.8.0p5* [WWW document]. URL ftp://ftp.lambda.moo.mud.org/pub/MOO/html/ProgrammersManual_toc.html 48
- David, J.M., Krivine J.P. & Simmons, R. (eds.) (1993) *Second Generation Expert Systems*. Springer-Verlag: Berlin. 5
- Dieberger, A. (1997) Supporting social navigation on the World Wide Web. *International Journal of Human-Computer Studies* 46, 805-825. 79
- Dieberger, A. and Tromp, J.G. (1993) The Information City Project - a virtual reality user interface for navigation in information spaces. Paper presented at the *Symposium Virtual Reality* Vienna, Dec 1-3., 1993. 74
- Dieng, R., Corby, O., Giboin, A. & Ribière, M. (1998) Methods and Tools for Corporate Knowledge Management. In *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'98)*, Banff, Canada, April 18-23, 1998. 25
- Domingue, J. (1998) Tadzebao and WebOnto: Discussing, Browsing and Editing Ontologies on the Web. In B.R. Gaines & M. Musen (eds.) *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'98)*, Banff, Canada, April 18-23, 1998. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW98/domingue/> 24, 26, 27
- Edwards, D.M. & Hardman, L. (1989) "Lost in hyperspace": cognitive mapping and navigation in a hypertext environment. In R. McAleese (ed.) *Hypertext: Theory into practice*. Intellect Books: Oxford. 105-125. 71
- Euzenat, J. (1996a) HyTropes: a WWW front-end to an object knowledge management system. In *Proceedings of the 10th Knowledge Acquisition Workshop (KAW'96)*. Banff, Canada, Nov. 9-14, 1996. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW96/euzenat/euzenat96c.html> 23, 27
- Euzenat, J. (1996b) Corporate memory through cooperative creation of knowledge bases and hyper-documents. In *Proceedings of the 10th Knowledge Acquisition Workshop (KAW'96)*. Banff, Canada, Nov. 9-14, 1996. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW96/euzenat/euzenat96b.html> 25, 26, 27

- Evans, R. (1993) Collaborative Networked Communication: MUDs as Systems Tools. In *Proceeds of the Seventh Systems Administration Conference (LISA VII)*, November 1993, Monterey, CA. 1-8. URL <http://www.ccs.neu.edu/home/remy/documents/cncmast.html> 42
- Farquhar, A., Fikes, R. & Rice, J. (1996) The Ontolingua Server: a Tool for Collaborative Ontology Construction. In *Proceedings of the 10th Knowledge Acquisition Workshop (KAW'96)*. Banff, Canada, Nov. 9-14, 1996. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW96/farquhar/farquhar.html> 22, 27, 31
- Farquhar, A., Fikes, R., Pratt, W. & Rice, J. (1995) *Collaborative Ontology Construction for Information Integration*. Research report 63, Knowledge System Laboratory, Stanford University, Stanford, CA, US. URL ftp://ksi.stanford.edu/pub/KSL_Reports/KSL-95-63.ps 22
- Feiner, S. (1988) Seeing the forest for the trees: hierarchical display of hypertext structures. In *Proceedings of the ACM Conference on Office Information Systems*, 205-212. 72
- Fensel, D. & Benjamins, V.R. (1996) Assumptions in model-based diagnosis. In *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'96)*. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW96/fensel/ambd.html> 8
- Fensel, D., Decker, S., Erdmann, M. & Studer, R. (1998) Ontobroker. Or How to Enable Intelligent Access to the WWW. In B.R. Gaines & M. Musen (eds.) *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'98)*, Banff, Canada, April 18-23, 1998. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW98/fensel/> 20, 211
- Finin, T., Labrou, Y. & Mayfield, J. (1997) KQML as an agent communication language. In J.M. Bradshaw (ed.) *Software Agents*. Cambridge, MA: AAAI/MIT Press. 196
- Fink, J., Kobsa, A. & Nill, A. (1997) Adaptable and Adaptive Information Access for All Users, Including the Disabled and the Elderly. In A. Jameson, C. Paris & C. Tasso (eds.) *User Modeling: Proceedings of the Sixth International Conference (UM97)*, Vienna, New York: Springer Wien New York. URL <http://um.org/> 84
- Fischer, G., Grudin, J., Lemke, A., McCall, R., Ostwald, J., Reeves, B. & Shipman, F. (1992) Supporting Indirect Collaborative Design With Integrated Knowledge-Based Design Environments. *Human-Computer Interaction* 7, 281-314. 36, 55, 66, 87
- Fischer, G., Lemke, A.C., McCall, R. & Morch, A.I. (1991) Making Argumentation Serve Design. *Human-Computer Interaction* 6, 393-419. 54, 55, 87
- Fisher, M.L. (1997) The TCE Corporate Technical Memory: groupware on the cheap. *International Journal of Human-Computer Studies* 46, 847-860. 24
- Fox, K.R. (1997) *MOOs with WWW gateways* [WWW document]. URL <http://www.ccs.neu.edu/home/fox/moo/www.html> 42, 49
- Fridman Noy, N. & Hafner, C.D. (1997) The State of the Art in Ontology Design. *AI Magazine Fall 1997*, 53-74. 7
- Gaines, B.R. & Shaw, M.L.G. (1987) Knowledge support systems. *ACM MCC University Research Symposium*, Austin, Texas, MCC, 47-66. 14, 18
- Gaines, B.R. & Shaw, M.L.G. (1996a) A Networked, Open Architecture Knowledge Management System. In *Proceedings of the 10th Knowledge Acquisition Workshop (KAW'96)*. Banff, Canada, Nov. 9-14, 1996. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW96/gaines/KM.html> 18, 101, 115, 214
- Gaines, B.R. & Shaw, M.L.G. (1996b) WebGrid: Knowledge Modeling and Inference through the World Wide Web. In *Proceedings of the 10th Knowledge Acquisition Workshop (KAW'96)*. Banff, Canada, Nov. 9-14, 1996. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW96/gaines/KMD.html> 18, 101, 214

- Gaines, B.R. & Shaw, M.L.G. (1998) Developing for Web Integration in Sisyphus-IV: WebGrid-II Experience. In B.R. Gaines & M. Musen (eds.) *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'98)*, Banff, Canada, April 18-23, 1998. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW98/gaines/> 18, 19, 103
- Gaines, B.R. (1987) Rapid prototyping for expert systems. In M. Oliff (ed.) *Intelligent Manufacturing: Proceedings from the First International Conference on Expert Systems and the Leading Edge in Production Planning and Control*, Menlo Park, California, 45-73. Benjamin Cummins.18
- Genesereth, M.R. & Fikes, R.E. (1992) *Knowledge Interchange Format, Version 3.0 Reference Manual*. Technical Report Logic-92-1, Computer Science Department, Stanford University. 196
- Gibbs, G. (1998) Modifying Multi-User Discussion systems to support text-based virtual learning environments on the Web - the coMentor experience. In R. Hazemi, S. Hailes & S. Wilbur (eds.) *The Digital University: Reinventing the Academy*. Berlin: Springer-Verlag. ISBN 1-85233-003-1. 115-134.95
- Girardin, L. (1996) Mapping the Virtual Geography of the World-Wide Web. In *Poster Proceedings of the Fifth International World Wide Web Conference (WWW5)*. May 6-10, Paris, France.71
- Greenhalgh, C. M. & Benford, S. D. (1995) MASSIVE: A Virtual Reality System for Teleconferencing. *ACM Transactions on Computer Human Interfaces (TOCHI)* 2(3), 239-261 39, 40
- Gruber, T. (1993) A translation approach to portable ontology specifications. *Knowledge Acquisition* 5(2), 199-220.29
- Gruber, T.R., Tenenbaum, J.M. & Weber, J.C. (1992) Toward a Knowledge Medium for Collaborative Product Development. In J.S. Gero (ed.) *Artificial Intelligence in Design '92. Proceedings of the Second International Conference in Design*. Pittsburgh, USA, June 22-25, 1992. 413-432. Kluwer Academic Publishers. URL <ftp://ksi.stanford.edu/pub/knowledge-sharing/papers/shade.ps>26
- Gruber, T.R., Vemuri, S. & Rice, J. (1997) Model-based virtual document generation. *International Journal of Human-Computer Studies* 46, 687-706. URL <http://www-ksi-svc.stanford.edu:5915/doc/papers/ksi-95-80>85
- Guarino, N. & Giaretta, P. (1995) Ontologies and knowledge bases – towards a terminological clarification. In N.J. Mars (ed.) *Towards Very Large Knowledge Bases – Knowledge Building and Knowledge Sharing 1995*, 25-32. IOS Press.6
- Guarino, N., Carrara, M. & Giaretta, P. (1994) An Ontology of Meta-Level Categories. In J. Doyle, E. Sandewall & P. Torasso (eds.) *Principles of Knowledge Representation and Reasoning: Proceedings of the 4th International Conference*. Morgan Kaufmann: San Mateo, CA. 198
- Hand, C. (1996) Some User Interface Issues for Hypermedia Virtual Environments. Position Paper for *Virtual Environments and the World-Wide Web Workshop at the Fifth International World-Wide Web Conference*, Paris, France.38
- Hardy, B.J., Doughty, S., Parretti, M., Tennison, J., Finn, B. & Gardner, K. (1998) Internet Conferences in Nuclear Magnetic Resonance Spectroscopy. *NMR Spectroscopy* 31(2/3). 47, 94
- Hardy, B.J., Doughty, S.W., Parretti, M.F., Richards, W.G. & Tennison, J. (1997) First Molecular Graphics and Modelling Society Electronic Conference. *Journal of Molecular Graphics and Modelling* 15, 141-144. 47, 94
- Hardy, B.J., Doughty, S.W., Parretti, M.F., Tennison, J. & Wilson, I. (1997) Internet Conferences in Glycobiology. *Glycobiology* 7, R9-R12 47, 94

- Hayes-Roth, F., Waterman, D.A. & Lenat, D.B. (1983) *Building Expert Systems*. Addison-Wesley: Reading, Mass.14
- Ibrahim, B. (1997) User of HTML forms in complex user interfaces for server-side applications. *International Journal of Human-Computer Studies* 46, 761-771.70
- ISO/IEC (1996) *Document Style Semantics and Specification Language (DSSSL)*. International Standard ISO/IEC 10179:1996. 205
- Jenkins, C., Jackson, M., Burden, P. & Wallis, J. (1998) Searching the world wide web: an evaluation of available tools and methodologies. *Information and Science Technology* 39, 985-994. 75, 76
- Jones, D., Bench-Capon, T. & Visser, P. (1998) Methodologies for Ontology Development. In J. Cuenca (ed.) *IT & KNOWS: Information Technology and Knowledge Systems. Proceedings of the XV IFIP World Computer Congress, 31 August - 4 September 1998, Vienna/Austria and Budapest/Hungary*..... 7, 198
- Kang, B.H., Compton P. & Preston, P. (1998) Simulated Expert Evaluation of Multiple Classification Ripple Down Rules. In B.R. Gaines & M. Musen (eds.) *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'98)*, Banff, Canada, April 18-23, 1998. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW98/kang/>18
- Karp, P.D. & Gruber, T. (1997) *The Generic Frame Protocol* [WWW document]. URL <http://www.ai.sri.com/~gfp/spec/paper/paper.html> 30, 95, 189
- Karp, P.D., Myers, K.L. & Gruber, T. (1995) The Generic Frame Protocol. In *Proceedings of the 1995 International Joint Conference on Artificial Intelligence (IJCAI'95)*, 768-774. URL <http://www.ai.sri.com/pubs/papers/Karp96-768:Generic/document.ps.Z>30
- Kelly, G.A. (1955) *The Psychology of Personal Constructs*. Norton: New York.12
- Kidd, A. (1994) The Marks are on the Knowledge Worker. In *Proceedings of ACM CHI'94: Human Factors in Computing Systems*. Boston, Mass, 24-28 April 1994. ACM Press: New York. 186-191.35
- Kollock, P. & Smith, M. (1996) Managing the Virtual Commons: Cooperation and Conflict in Computer Communities. In S. Herring (ed.) *Computer-Mediated Communication: Linguistic, Social and Cross-Cultural Perspectives*. Amsterdam: John Benjamins. 109-128. URL <http://netscan.sscnet.ucla.edu/csoc/papers/virtcomm/>.....36
- Kollock, P. (1996) Design Principles for Online Communities. In *Proceedings of Harvard Conference on the Internet and Society*. Also (1998) *PC Update* 15(5), 58-60. URL <http://www.sscnet.ucla.edu/soc/faculty/kollock/papers/design.htm>47
- Kraut, R., Galegher, J., Fish, R. & Chalfonte, B. (1992) Task Requirements and Media Choice in Collaborative Writing. *Human-Computer Interaction* 7, 275-407..... 39, 40, 41
- Lee, J. & Lai, K-Y. (1991) What's in Design Rationale? *Human-Computer Interaction* 6, 251-280.....56
- Lee, J. (1990) SIBYL: A qualitative design management system. In P.H. Winston & S. Shellard (eds.) *Artificial intelligence at MIT: Expanding frontiers*. MIT Press: Cambridge, MA. 104-133.....56
- Lenat, D. (1998) *The Dimensions of Context Space*. [WWW document] URL <http://www.cyc.com/context-space.doc>9
- Lin, J. (1994) Information Sharing and Knowledge Merging in Cooperative Information Systems. In *Proceedings of the Fourth Workshop on Information Technologies and Systems*, Vancouver. 58-66. 199
- Lindop, L., Sriskandarajah, M., Williams, M., Bracken, M., Cadden, N., Dabbs, A. & Gallagher, W. (1997) Catching sites. *PC Magazine* 6(2), 109-153.....75

- Lynch, P.J. (1995) *Web Style Manual* [WWW document]. URL <http://info.med.yale.edu/caim/Manual-1.html>67
- MacLean, A. Bellotti, V. & Shum, S. (1993) Developing the Design Space with Design Space Analysis. In P.F. Byerley, P.J. Bamard & J. May (eds.) *Computers, Communication and Usability: Design issues, research and methods for integrated services*, Chapter 2.4, 197-219. (North Holland Series in Telecommunications) Elsevier: Amsterdam.....58
- MacLean, A., Young, R.M., Bellotti, V.M.E. & Moran, T.P. (1991) Questions, Options and Criteria: Elements of Design Space Analysis. *Human-Computer Interaction* 6, 201-250..... 51, 58
- Mahalingam, K. & Huhns, M.N. (1997) A Tool for Organizing Web Information. *Computer June 1997*, 80-83.24
- Mahesh, K. & Nirenburg, S. (1995) A Situated Ontology for Practical NLP. *IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, 19-20 August, 1995. 198
- Mahesh, K. (1996) *Ontology Development for Machine Translation: Ideology and Methodology*. Technical Report MCCA 96-292, Computing Research Laboratory, New Mexico State University, Las Cruces, NM. 198
- Marchiori, M. (1997) The Quest for Correct Information on the Web: Hyper Search Engines. *Computer Networks and ISDN Systems: Proceedings of the Sixth International World Wide Web Conference 29(8-13)*, 1225-1235.77
- Markus, M.L. (1994) Finding a Happy Medium: Explaining the Negative Effects of Electronic Communication on Social Life at Work. *ACM Transactions on Information Systems* 12(2), 119-149.35
- Mars, N.J.I., ter Stal, W.G., de Jong, H., van der Vet, P.E. & Speel, P.-H. (1994) Semi-automatic Knowledge Acquisition in Plinius: An Engineering Approach. In *Proceedings of the 8th Banff Knowledge Acquisition for Knowledge-based Systems Workshop*, Banff, 30 January - 4 February, 1994. 198
- Marshall, C.C. & Rogers, R.A. (1992) Two Years before the Mist: Experiences with Aquanet. In *Proceedings of the ACM ECHT Conference*, Milan, Italy, 30 November - 4 December, 1992. 53-62.65
- Marshall, C.C., Halasz, F.G., Rogers, R.A. & Janssen Jr., W.C. (1991) Aquanet: a hypertext tool to hold your knowledge in place. In *Proceedings of Hypertext'91*, San Antonio, TX, 16-18 December, 1991. 261-275. 65, 189
- Masinter, L. & Ostrom E. (1993) Collaborative Information Retrieval: Gopher from MOO. In *Proceedings of INET'93*.....78
- McCarthy, J.C. & Monk, A.F. (1994) Measuring the quality of computer-mediated communication. *Behaviour & Information Technology* 13(5), 311-319..... 35, 42
- McCarthy, J.C., Miles, V.C., Monk, A.F., Harrison, M.D., Dix, A.J. & Wright, P.C. (1993) Text-Based On-Line Conferencing: A Conceptual and Empirical Analysis Using a Minimal Prototype. *Human-Computer Interaction* 8, 147-183..... 39, 41, 45
- McGraw, K. & Harbison-Briggs, K. (1989) *Knowledge Acquisition: Principles and Guidelines*. Englewood Cliffs, NJ: Prentice-Hall.....15
- Neal, L. & Mantei, M. (1993) Computer-Supported Meeting Environments. In *Tutorial Notes of Conference on Human Factors in Computing Systems (INTERCHI'93)*, Amsterdam, The Netherlands.15
- Newell, A. (1982) The Knowledge Level. *Artificial Intelligence* 18, 87-127.....5
- Nicol, D., Smeaton, C. Slater, A.F. (1995) Footsteps: Trail-blazing the Web. *Computer Networks and ISDN Systems* 27, 879-885.67

- O'Hara, K., Shadbolt, N.R. & van Heijst, G. (in press) Generalised Directive Models: integrating model development and knowledge acquisition. *International Journal of Human-Computer Studies*.....18
- Olaniran, B.A. (1996) A model of group satisfaction in computer-mediated communication and face-to-face meetings. *Behaviour & Information Technology* 15(1), 24-36..... 35, 46
- O'Leary, D.E. (1997) Impediments in the use of explicit ontologies for KBS development. *International Journal of Human-Computer Studies* 46, 327-337.....15
- Olsen, K.A., Korfhage, R.R., Sochats, K.M., Spring, M.B. & Williams, J.G. (1993) Visualisation of a Document Collection: The VIBE System. *Information Processing and Management* 29(1), 69-81. 37, 73
- Pringle, G., Allison, L. & Dowe, D.L. (1998) What is a tall poppy among Web pages? *Computer Networks and ISDN Systems* 30(1-7), 369-377.....76
- Reid, F.J.M., Malinek, V., Stott, C.J.T. & Evans, J.S.T.B.T. (1996) The messaging threshold in computer-mediated communication. *Ergonomics* 39(8), 1017-1037..... 39, 44, 47
- Rein, G.L. & Ellis, C.A. (1991) rIBIS: A Real-Time Group Hypertext System. *International Journal of Man-Machine Studies* 24(3), 349-367.52
- Rice, J., Farquhar, A., Piernot, P. & Gruber, T. (1996) Using the Web Instead of a Window System. In *Proceedings of Computer-Human Interaction '96 (CHI'96)*, Vancouver, BC, Canada, April 13-18, 1996. 103 - 110. URL <http://www-ksi-svc.stanford.edu:5915/doc/papers/ksi-95-69/ksi-95-69-linearised.html>.....22
- Rittel, H. & Kunz, W. *Issues as elements of information systems*. Working paper #131, Institute of Urban and Regional Development, University of California, Berkeley and Technical Report S-78-2, Institut für Grundlagen der Planung I.A., University of Stuttgart.51
- Rittel, H.W.J. & Webber, M.M. (1973) Dilemmas in a General Theory of Planning. *Policy Sciences* 4, 155-169.....51
- Rittel, H.W.J. (1972) Second Generation Design Methods. Reprinted in Cross, N. (ed.) (1984) *Developments in Design Methodology*, 317-327. J. Wiley & Sons: Chichester.....51
- Schneider, D.K. (1997) *Educational Technology: Educational VR (MUD) sub-page* [WWW document]. URL <http://tecfa.unige.ch/edu-comp/WWW-VL/eduVR-page.html>42
- Schreiber, A.T., Wielinga, B.J., Akkermans, J.M., Van de Velde, W. & Anjewierden, A. (1994) CML: the CommonKADS conceptual modelling language. In L. Steels, A.T. Schreiber & W. Van de Velde (eds.) *A Future for Knowledge Acquisition. Proceedings of the 8th European Knowledge Acquisition Workshop (EKAW'94)*, Berlin: Springer-Verlag. 1-25.29
- Schreiber, A.Th., Wielinga, B.J. & Jansweijer, W.H. (1995) The KACTUS View on the 'O' Word. *IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, 19-20 August, 1995..... 198
- Shadbolt, N. & Burton, M. (1995) Knowledge elicitation: a systematic approach. In J.R. Wilson & E.N. Corlett (eds.) *Evaluation of Human Work: A practical ergonomics methodology*. Second Edition. Taylor & Francis: UK.10
- Shadbolt, N. (1989) Expert systems - a natural history. In N. Shadbolt (ed.) *Research and Development in Expert Systems VI. Proceedings of Expert Systems 89, the Ninth Annual Technical Conference of the British Computer Society Specialist Group on Expert Systems*, London, 20-22 September 1989. Cambridge University Press: Cambridge, UK. 1-11.5
- Shadbolt, N.R. & Burton, A.M. (1989) Empirical studies in knowledge elicitation. *ACM-SIGART Special Issue on Knowledge Acquisition*, 108.10

- Shaw, M.L.G. & Gaines, B.R. (1989) Comparing conceptual structures: consensus, conflict, correspondence and contrast. *Knowledge Acquisition* 1, 341-363. 16, 114, 199, 214
- Shaw, M.L.G. & Gaines, B.R. (1998) WebGrid II: Developing Hierarchical Knowledge Structures from Flat Grids. In B.R. Gaines & M. Musen (eds.) *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'98)*, Banff, Canada, April 18-23, 1998. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW98/shaw/> 14, 18, 105
- Shipman, F.M. & McCall, R.J. (1997) Integrating Different Perspectives on Design Rationale: Supporting the Emergence of Design Rationale from Design Communication. *Artificial Intelligence in Engineering Design, Analysis and Manufacturing (AIEDAM)* 11(2), 141-154.....50
- Shneiderman, B. (1982) The future of interactive systems and the emergence of direct manipulation. *Behaviour and Information Technology*, 1, 237-256.....49
- Shneiderman, B. (1983) Direct manipulation: A step beyond programming languages. *IEEE Computer*, 16, 57-69.49
- Shum, S. (1993) *QOC Design Rationale Retrieval: A Cognitive Task Analysis, & Design Implications*. Technical Report, Rank Xerox EuroPARC, Cambridge, UK.....59
- Suke, D. (1995) Conventions for Reaching Agreement on Shared Ontologies. In *Proceedings of the 9th Knowledge Acquisition Workshop (KAW'95)*, Banff, Canada, Feb-March, 1995.....8, 26
- Smith, P.A. & Wilson, J.R. (1993) Navigation in hypertext through virtual environments. *Applied Ergonomics* 24(4), 271-278. 47, 71, 72, 74
- Smith, P.A. Newman, I.A. & Parks, L.M. (1997) Virtual hierarchies and virtual networks: some lessons from hypermedia usability research applied to the World Wide Web. *International Journal of Human-Computer Studies* 47, 67-95.....75
- Sproull, L. & Kiesler, S. (1986) Reducing social context cues: Electronic mail in organizational communication. *Management Science* 32(11), 1492-1512.....47
- St. Laurent, S. (ed.) (1998) *XSchema*. [WWW Document] URL <http://purl.oclc.org/NET/xschema/> 203
- Steels, L. (1993) Corporate knowledge management. In *Proceedings of ISMICK'93*, Compiègne, France, 9-30.....25
- Swartout, B., Patil, R., Knight, K. & Russ, T. (1996) Toward Distributed Use of Large-Scale Ontologies. In *Proceedings of the 10th Knowledge Acquisition Workshop (KAW'96)*. Banff, Canada, Nov. 9-14, 1996. URL http://ksi.cpsc.ucalgary.ca/KAW/KAW96/swartout/Banff_96_final_2.html 8, 21, 22, 27, 28
- Tauscher, L. & Greenberg, S. (1997) How people revisit web pages: empirical findings and implications for the design of history systems. *International Journal of Human-Computer Studies* 47, 97-137. 70, 71
- Toulmin, S. (1958) *The Uses of Argument*. Cambridge University Press: Cambridge.....50
- Trigano, P. (1994) Automatic Indexation and Knowledge Storing. In *Proceedings of ISMICK'94*, Compiègne, France, 223-235.....25
- Tromp, J.G. (1993) *Results of two surveys about Spatial Perception and Navigation of a Text-Based Spatial Interface* [WWW document]. URL <http://www.cms.dmu.ac.uk/~cph/VR/JolaPaper/jola.html>39
- Twidale, M.B., Nichols, D.M. & Paice, C.D. (1996) *Browsing is a Collaborative Process*. Technical Report CSEG/1/96. Computing Department, Lancaster University. URL <http://www.comp.lancs.ac.uk/computing/research/cseg/projects/ariadne/docs/bcp.html> 67, 77

- Uschold, M. (1998) Knowledge level modelling: concepts and terminology. *The knowledge engineering review* 13(1), 5-29.....28
- Uschold, M., Clark, P., Healy, M., Williamson, K. & Woods, S. (1998) An Experiment in Ontology Reuse. In B.R. Gaines & M. Musen (eds.) *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'98)*, Banff, Canada, April 18-23, 1998. URL <http://ksi.cpsc.ucalgary.ca/KAW/KAW98/uschold/>.....31
- van Heijst, G., Schreiber, A.Th. & Wielinga, B.J. (1997) Using explicit ontologies in KBS development. *International Journal of Human-Computer Studies* 45, 183-292.6, 24
- Warfield, J.N. (1971) *Societal Systems: Planning, Policy and Complexity*. Wiley: New York.15
- Wielinga, B.J., Schreiber, A.Th., Jansweijer, W.H., Anjewierden, A. & van Harmelen, F. (1994) *Framework and Formalism for Expressing Ontologies*. KACTUS Project Deliverable DO1b.1, University of Amsterdam.198
- Wilson, A. (1994) *tkMOO-light for UNIX, Windows and Macintosh* [WWW document]. URL <http://www.cm.cf.ac.uk/User/Andrew.Wilson/tkMOO-light>49
- Winograd, T. (1988) A Language/Action Perspective on the Design of Cooperative Work. *Human-Computer Interaction* 3, 3-30.....41
- Wittenburg, K., Das, D., Hill, W. & Stead, L. (1995) Group Asynchronous Browsing on the World Wide Web. In *Proceedings of the Fourth International World Wide Web Conference*, Boston, MA.77
- Wolf, H. & Froitzheim, K. (1998) User space meets document space. *Computer Networks and ISDN Systems* 30(1-7), 710-712.78
- Wood, A.M., Drew, N.S., Beale, R. & Hendley, R.J. (1995) HyperSpace: Web Browsing with Visualisation. In R. Holzapfel (ed.) *Poster Proceedings of the Third International WWW Conference: Technology, Tools and Applications*. April 10-14, Darmstadt, Germany. URL <http://www.igd.fhg.de/www/www95/proceedings/posters/35/index.html>71
- World Wide Web Consortium (1996) *Cascading Style Sheets, level 1*. W3C Recommendation. [WWW Document] URL <http://www.w3c.org/TR/REC-CSS1>205
- World Wide Web Consortium (1997) *HTML 3.2 Recommendation*. [WWW Document] URL <http://www.w3c.org/TR/>.....97
- World Wide Web Consortium (1998a) *Extensible Markup Language (XML) 1.0*. W3C Recommendation. [WWW Document] URL <http://www.w3c.org/TR/REC-xml>.....201
- World Wide Web Consortium (1998b) *Cascading Style Sheets, level 2: CSS2 Specification*. W3C Recommendation. [WWW Document] URL <http://www.w3c.org/TR/REC-CSS2>.....205
- World Wide Web Consortium (1998c) *Extensible Stylesheet Language*. W3C Working Draft. [WWW Document] URL <http://www.w3c.org/TR/WD-xsl>.....205
- World Wide Web Consortium (1998d) *XML Linking Language (XLink)*. W3C Working Draft. [WWW document] URL <http://www.w3c.org/TR/WD-xlink>208
- World Wide Web Consortium (1998e) *XML Pointer Language (XPointer)*. W3C Working Draft. [WWW Document] URL <http://www.w3c.org/TR/WD-xptr>.....208
- World Wide Web Consortium (1998f) *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Working Draft. [WWW Document] URL <http://www.w3c.org/TR/WD-rdf-syntax>.....210

APPENDIX I APECKS README FILE

The following file and those it references are included on the CD-ROM accompanying this thesis.

README

CONTENTS

This CD contains the following directories and files:

- | | |
|-----------------------------|---|
| 1. README.txt | -- This file |
| 2. APECKScore.db | -- The APECKS MOO-database file |
| 3. LambdaMOO-1.8.0r5.tar.gz | -- The Lambda MOO source files |
| 4. extensions.c | -- Command extensions necessary to run APECKS |
| 5. icons.tar.gz | -- icons for use with APECKS |

LambdaMOO-1.8.0r5 is the latest version of the MOO server at time of writing. You may wish to download a more recent version: you can get one by following links from <http://www.moo.mud.org/>.

This document gives information on the following:

- * Requirements
- * Installation
- * Starting the MOO
- * Connecting to the MOO
- * Connecting by WWW
- * Viewing APECKS Code
- * Getting Help
- * Acknowledgements

REQUIREMENTS

To run APECKS, you usually need a unix-based machine. MOOs can be run on Win-NT: you should search for information about doing so within the MOO-Cows-FAQ from <http://www.moo.mud.org/>.

INSTALLATION

To install APECKS, follow these steps:

1. Move all the contents of this CD to a directory on your unix system
2. Gunzip and untar the MOO server with the following commands:


```
% gunzip LambdaMOO-1.8.0r5.tar.gz
% tar -xvf LambdaMOO-1.8.0r5.tar
```
3. Copy extensions.c from the CD into the MOO server directory, overwriting the copy that is already there (you may wish to make a backup of that copy first).


```
% cp extensions.c MOO-1.8.0r5/
```
4. Compile the MOO server for your machine by following the instructions given in MOO-1.8.0r5/readme
5. Move the MOO server (MOO-1.8.0r5/moo) and the restart script (MOO-1.8.0r5/restart) into a directory with the APECKS core database.
6. Gunzip and untar the icons with the following commands:


```
% gunzip icons.tar.gz
% tar -xvf icons.tar
```

7. Move or copy APECKSicons into a WWW-accessible location, such as your public_html directory.

STARTING THE MOO

Before you begin the MOO, you must decide on the ports on which the MOO will run. You need to decide on two ports, one for the telnet (normal) interface to the MOO and one for the WWW interface.

Port numbers on unix systems are random numbers from 1 to 9999, so there is plenty of choice, but there are two limitations. Firstly, you cannot choose a port below 1024 unless you are running as root (which, for security reasons, you probably should not be). Secondly, you cannot choose a port that is already being listened to by another server application.

For the purposes of this example, I will use the ports 8888 for the telnet port and 8080 for the WWW port.

To start the MOO, move to the directory with the MOO server, restart script and APECKS core database in it, and run the restart script:

```
% ./restart APECKScore 8888
```

The load will take a few minutes. You can keep an eye on how it's progressing by periodically having a look at the log file:

```
% tail APECKScore.log
```

CONNECTING TO THE MOO

You can connect to the MOO using a MOO client or through telnet. Make sure you specify the port that you started the MOO on:

```
% telnet <machine.name> 8888
```

When you connect, you will see a welcome message and be prompted to connect with the username Wizard and no password. The command for connecting is:

```
> connect Wizard
```

or, more generally:

```
> connect <username> <password>
```

Various other frequent commands are:

```
@quit      -- quit the MOO
help       -- get help on the MOO
@who       -- see who else is connected
look       -- look at the room you're in
"<message>" -- say something
```

The first time you connect to the MOO is special, and there are a number of things that you should do after connecting.

1. Make yourself an avatar apart from Wizard: you should never connect as Wizard in the normal running of things.
 - a. Make the avatar using the command:


```
@make-player <name> <email>
```
 - b. Change the password of the avatar using the command:


```
@new-password <name> is <password>
```
 - c. Give the avatar programmer status using the command:


```
@programmer <name>
```
 - d. Find the object number of the avatar using the command:


```
#<name>
```
 - e. Use the object number to give the avatar wizard status:


```
; <avatarobj>.wiz = 1
```
2. Disconnect as wizard and reconnect as the avatar you have just created. Then

change the password of Wizard so that no one can connect with that much power:

```
> ; #2.password = "Something impossible to type"
```

- Now you can start customising the MOO. The only things that are essential to do at this point is to change the properties on \$network as appropriate. To look at the properties, do:

```
> @display $network.
```

For each property, set it as appropriate using the command:

```
; $network.<property> = <new value>
```

- The next vital thing to do is to specify the WWW port on which the MOO should be listening:

```
> ; unlisten($www_utils.www_port)
> ; $www_utils.www_port = 8080
> ; listen($web_system, $www_utils.www_port, 0)
```

- Finally, to ensure that, when you connect through the WWW, you do not get missing images, point the MOO at the location in which you placed icons APECKS uses:

```
> ; $www_utils.icon_server = "<your url>/APECKSicons/"
```

You can see which icons APECKS refers to using the command:

```
@icons $www_utils
```

You can change the icon that APECKS uses for a particular purpose with the command:

```
@chicon $www_utils as <purpose> at <URL>
```

See help \$www_utils:@icons and help \$www_utils:@chicon for details.

CONNECTING BY WWW

You should now be able to connect to the MOO through the WWW as your avatar. The URL is:

```
http://<machine.name>:8080/
```

You will be prompted for your name and password: they are the same as those you use to connect using telnet.

To view the collaborative ontology-construction support within APECKS, go to the URL:

```
http://<machine.name>:8080/$domain/domains
```

From there, you will be able to create your own domains and roles. To allow other people to view them and create their own roles, you must create avatars for them by connecting to the MOO via telnet and using the commands:

```
> @make-player <username> <email>
> @new-password <username> is <password>
```

You should not make these users wizards unless you trust them.

VIEWING APECKS CODE

The easiest way to view the APECKS code is when connected to the MOO through telnet.

Although the database is in textual format, it is not designed to be human-readable.

You have to be a programmer status within the MOO in order to view code.
 Wizards
 can give people programmer status with the command:

```
> @programmer <username>
```

The following commands are helpful for viewing code:

```
@props <object>      -- lists the names of the properties on an object
@verbs <object>       -- lists the names of the verbs on an object
@display <object>.    -- lists details about the object properties
@display <object>:    -- lists details about the object verbs
@list <object>:<verb> -- lists the verb code
```

Further help on viewing code is available within the MOO: try help prog-index as a starting point, or help <command> for any of the above commands.

The objects within the MOO that are novel to APECKS and completely or substantially written by me are as follows (asterisked objects are those of particular note):

WWW-related objects:

```
* $web_system
* $web_request
* $web_server_options
  $web_root
  $www_utils
  $html_utils
  $web_options
  $imap_agent
```

Mirroring objects:

```
* $mirror
* $mirror_link
```

Ontology objects:

```
* $relation
* $domain
* $role
* $frame
* $individual
* $class
* $slot
  $facet
* $comparison
* $criteria
* $argument
  $frs_utils
  $keyword_db
  $role_db
  $class_db
  $individual_db
  $slot_db
  $facet_db
  $criteria_db
  $annotation_db
  $apecks_options
```

Experimenting objects:

```
$subject
$logging_web_request
```

Real-life players:

```
$rlpc
$rlpc_help
```

Aside from those specified on the above objects, the following verbs are vital to the APECKS system:

```
Retrieving structure and components
<all objects>:www_direct
```



```
#1:www_retrieve
```

```
Determining structure
```

```
<all objects>:www_structure
<all objects>:www_<page name>
```

```
Generating components
```

```
<all objects>:html_<component name>
```

```
Generating forms for changing objects
```

```
#1:obvious_web_verbs
<all objects>:<action>_web
```

```
Caching page components
```

```
#1:update_cache
#1:clear_caches
#1:set_www_*
#1:@clear-caches
```

GETTING HELP

```
-----
```

APECKS is not as well documented as it probably should be.

Further information about MOOs, MOOing, MOO clients, and the MOO programming language can be found from <<http://www.moo.mud.org/>>.

Problems specifically with APECKS itself, queries about adapting APECKS for your purposes and other APECKS-related questions should be addressed to me, Jeni Tennison at <jeni@friday.u-net.com>.

ACKNOWLEDGEMENTS

```
-----
```

APECKS was developed as part of research for Jeni Tennison's PhD at the University of Nottingham, UK, and was supported financially by the School of Psychology, University of Nottingham.

APECKS is based on the LambdaMOO database, which was developed by numerous MOOers over the years, and on the WOOM database, which was developed by several people including Jeni Tennison, Kristian Fuglevik and Mark Bowyer.

APPENDIX II BACKGROUND QUESTIONNAIRE

This questionnaire looks at your background, both concerning 'mammals' and your computer experience.

There are three types of questions. In questions with boxes (☐) , you should place a tick in the relevant box (☐). In questions with dotted lines, you should write your answer in the space provided.

Some of the questions below ask you to rate your experience or interest on a scale similar to this one:

No experience /----/----/----/----/----/----/ A lot of experience

Rate your experience or interest by marking a cross within one of the sections on the scale. For example, if you have very little experience, you should mark the scale:

No experience /----/-X--/----/----/----/----/ A lot of experience

Please answer as many of the following questions as possible.

Background

1. What is your highest qualification in biology?

- ☐ GCSE/'O' level ☐ 'A'/'AS' level ☐ Bachelors Degree
☐ Masters Degree ☐ Doctorate

2. What is your current position?

- ☐ Undergraduate Student ☐ Postgraduate Student
☐ Postdoc/Lecturer ☐ Senior Lecturer/Professor

3. What is your specialisation (e.g. research area or favourite module) within the field of biology?

.....

4. How long have you been learning or working in the area of biology?

.....

Computer Experience

For the following questions, please rate your experience with the technology named.

5. Email

No experience /----/----/----/----/----/----/ A lot of experience

6. World Wide Web (WWW)

No experience /----/----/----/----/----/----/ A lot of experience

7. Chatrooms / MUDs / MOOs / Collaborative Virtual Environments

No experience /----/----/----/----/----/----/ A lot of experience

Domain & Task Experience

For the following questions, please rate your experience with the topic area named.

8. Mammals

No experience /----/----/----/----/----/----/ A lot of experience

9. Morphology (what animals look like)

No experience /----/----/----/----/----/----/ A lot of experience

10. Diet (what animals eat)

No experience /----/----/----/----/----/----/ A lot of experience

11. Habitat (where animals live)

No experience /----/----/----/----/----/----/ A lot of experience

Areas of Interest

12. What do you think is the most interesting aspect of the study of mammals?

.....

.....

13. How interesting would you rate that aspect as?

Not at all interesting /----/----/----/----/----/----/ Extremely interesting

14. What aspect of the study of mammals are you most experienced in?

.....

.....

15. How experienced would you say you are in that aspect?

Not at all experienced /----/----/----/----/----/----/ Extremely experienced

APPENDIX III USABILITY QUESTIONNAIRE

This questionnaire looks at how easy you found it to carry out different types of action within APECKS (the system you have been using).

Each section deals with a different kind of activity. There are then four questions to answer for each activity. The first question asks you to rate the ease of carrying out the activity within APECKS on a scale similar to this one:

Extremely difficult /----/----/----/----/----/----/----/ Extremely easy
Rate how easy it was to carry out the activity by marking a cross within one of the sections on the scale. For example, if it was very difficult, you should mark the scale:

Extremely difficult /----/-X--/----/----/----/----/----/ Extremely easy
The following three questions ask you what the easiest and most difficult aspects of carrying out the activity were, and for any other comments you have about using APECKS for that activity.

Please answer as many of the following questions as you can, in as much detail as you can. If you have not carried out an activity, try to guess what it would be like.

Finding Information

1. Rate the ease of finding information within APECKS.

Extremely difficult /----/----/----/----/----/----/----/ Extremely easy

2. What was the most difficult aspect of finding information within APECKS?

3. What was the easiest aspect of finding information within APECKS?

4. Have you any other comments about finding information within APECKS?

Adding Information

5. Rate the ease of adding information to APECKS.

Extremely difficult /----/----/----/----/----/----/----/ Extremely easy

6. What was the most difficult aspect of adding information to APECKS?

7. What was the easiest aspect of putting adding information to APECKS?

8. Have you any other comments about putting adding information to APECKS?

Changing Information

9. Rate the ease of changing information within APECKS.

Extremely difficult /----/----/----/----/----/----/----/ Extremely easy

10. What was the most difficult aspect of changing information within APECKS?

11. What was the easiest aspect of changing information within APECKS?

12. Have you any other comments about changing information within APECKS?

Removing Information

13. Rate the ease of removing information from APECKS.

Extremely difficult /---/---/---/---/---/---/---/ Extremely easy

14. What was the most difficult aspect of removing information from APECKS?

15. What was the easiest aspect of removing information from APECKS?

16. Have you any other comments about removing information from APECKS?

Comparing Roles

17. Rate the ease of comparing roles within APECKS.

Extremely difficult /---/---/---/---/---/---/---/ Extremely easy

18. What was the most difficult aspect of comparing roles within APECKS?

19. What was the easiest aspect of comparing roles within APECKS?

20. Have you any other comments about comparing roles within APECKS?

Discussing

21. Rate the ease of discussing information with others using APECKS.

Extremely difficult /----/----/----/----/----/----/----/ Extremely easy

22. What was the most difficult aspect of discussing information with others using APECKS?

.....

.....

.....

23. What was the easiest aspect of discussing information with others using APECKS?

.....

.....

.....

24. Have you any other comments about discussing information with others using APECKS?

.....

.....

.....

General

25. Have you any other comments about using APECKS that you wish to make?

.....

.....

.....

.....

.....

APPENDIX IV ONTOLOGY QUESTIONNAIRE

The following questionnaire involves rating an ontology by indicating how much you agree with a number of statements on scales like this:

Disagree completely /----/----/----/----/----/----/----/ Agree completely
 Rate how much you agree with the statement by marking a cross within one of the sections on the scale. For example, if you have strongly disagree, you should mark the scale:

Disagree completely /----/-X--/----/----/----/----/----/ Agree completely
 Please answer as many of the following questions as possible.

-
1. The ontology is detailed.
 Disagree completely /----/----/----/----/----/----/----/ Agree completely
 2. The ontology is imprecise.
 Disagree completely /----/----/----/----/----/----/----/ Agree completely
 3. The ontology is narrow.
 Disagree completely /----/----/----/----/----/----/----/ Agree completely
 4. The ontology is consistent within itself.
 Disagree completely /----/----/----/----/----/----/----/ Agree completely
 5. The ontology is incomplete.
 Disagree completely /----/----/----/----/----/----/----/ Agree completely
 6. The ontology is readable.
 Disagree completely /----/----/----/----/----/----/----/ Agree completely
 7. The ontology has not been the source of discussion.
 Disagree completely /----/----/----/----/----/----/----/ Agree completely
 8. The ontology is useful.
 Disagree completely /----/----/----/----/----/----/----/ Agree completely

APPENDIX V VOCABULARY LIST

The following alphabetical vocabulary list gives an explanation for the terms that are used within APECKS (the system you are using). Italics within an explanation indicate another term on the list.

The examples within the vocabulary list are taken from the *domain* 'film stars'.

Annotation

An annotation is a note or comment about something, usually asking about or explaining some information.

Cardinality

The cardinality is the number of *values* that a *slot* can hold at a time for an *individual*. The minimum cardinality is the smallest number of values that can be taken, and the maximum cardinality is the largest number of values that can be taken. For example, the slot 'channels', that recorded which channels a TV star appears on, has a minimum cardinality of '1', because every TV star has to appear on some channel, and a maximum cardinality of '5', because there are five (terrestrial) channels.

Class

A class is a group of *individuals*. Individuals within a class are called 'members' of the class. The classes that an individual belongs to are called its 'types'. For example, the members of the class 'weathermen' might include the individuals 'John Kettley', 'Ulrika Johnson' and 'Michael Fish'. The types of the individual 'John Kettley' might include the classes 'weathermen', 'men' and 'besuited'.

Comparison

A comparison is a description of the differences between two *roles*. It shows how the record of one person's opinion and knowledge differs from another person's.

Criterion

A criterion is a short statement of a rule used to decide on how things are best recorded. For example, there might be a criterion that says 'Technical terminology should be used wherever possible.' Another criterion might say 'Technical terminology should be avoided at all costs.' Criteria can contradict each other, in which case it is up to you to decide which rule to follow.

Default Value

The default value is the *value* that all *individuals* in a *class* have for a *slot* by default. For example, all the members of the class 'entertainer' might have the default value 'true' for the slot 'can sing.'

Domain

A domain is an area of knowledge. In this experiment, the domain is mammals.

Individual

An individual is a thing within a *domain*. In this experiment, the individuals are mammals, like 'sheep' and 'lion', and other biology-related concepts, like physical features, eating habits and environments.

Inverse

The inverse of a slot is a *slot* which expresses the opposite relationship. For example, the inverse of a slot 'mother' would be the slot 'child' and vice versa.

Range

The range gives the constraints on the types of *values* that a *slot* can hold for an *individual* or members of a *class*. For example, the range of the slot 'gender' for members of the class 'people' is the two categories 'male' and 'female'. For the class 'women', the range is just 'female'.

Repertory Grid

A repertory grid is a way of exploring and finding out more about what you know and think about a *domain*. It is also used to compare your *roles* with someone else's.

Representation

A representation is another term for a record of some information.

Role

A role is a record of one person's knowledge and thoughts about a *domain*.

Slot

A slot is a way of holding information about an *individual*. Slots have different *values* for different individuals. Sometimes, all individuals in a *class* have the same value - this is the *default value*. For example, TV stars might have slots like 'number of shows', 'speciality' or 'copresenters'.

Subclass partition

A subclass partition is a set of subclasses of a *class* where any *individual* will only be a member of one of the subclasses. If the subclass partition is exhaustive, this means that every individual within the class must be a member of one of the subclasses in the subclass partition. For example, an exhaustive subclass partition of the class 'TV stars' might be the set of subclasses 'men' and 'women' - TV stars cannot be both men and women (which makes it a subclass partition), and all TV stars must be one or the other (which makes it exhaustive).

Value

A value can be a number, a word, a phrase or a reference to another *individual*. For example, the *slot* 'number of shows' might hold numerical values; the slot 'speciality' might hold text describing the speciality; and the slot 'copresenters' might hold references to other TV stars.

APPENDIX VI INSTRUCTIONS – FIRST SESSION

We are building a computer-based ('Mammal-Spotter's Guide' that will help people identify any mammals they see / guide for zoos about what to feed the mammals in their care / tour guide that will help people learn about what mammals they are likely to encounter when they go on holiday).

In order to find out information for the ('Mammal-Spotter's Guide' / feeding guide / tour guide), we're asking you to use a system called APECKS, which is supposed to be good at getting at people's knowledge. As you use the system, you'll build a record of what you know. **Because the record is going to be used by other people, you should try to make things as clear as possible.**

Knowledge engineers have developed ways of recording knowledge, and during these sessions, you are going to be using one of these methods. This method involves having objects recording information about different aspects of an area of knowledge. **Have a look at the Vocabulary List and read the definitions for Role, Individual, Class and Slot. You can refer to the vocabulary list throughout the sessions.**

Everything that you do using this system will be logged for experimental use.

The system you are going to be using can be used through the World Wide Web, so you are going to be using Netscape. **Please do not enter new URLs and please do not use the 'Back' button.**

At the top of every screen is the **status bar**. On the far left, it tells you what kind of page you are looking at: a role, individual, class, slot and so on. The rest of the bar gives links to other objects, usually the domain and role, and sometimes a class.

Underneath the status bar is the **button bar**. The button bar gives links to other information about this object and lets you carry out some actions. Different kinds of objects have different buttons. **Feel free to explore what the different buttons do, but don't use the Recycle button unless you're sure you want to delete an object.**

The 'Help' button will give you help from any page.

The 'Actions' button takes you to a list of prompts which you can use to help you decide what to do next. There are 'Actions' buttons on Roles and on Classes.

Remember that we want to use this information for a ('Mammal-Spotter's Guide' / feeding guide / tour guide).

Please try to express what you think and know about mammals in as much detail and as clearly as possible.

APPENDIX VII INSTRUCTIONS – LATER SESSIONS

We are building a computer-based ('Mammal-Spotter's Guide' that will help people identify any mammals they see / guide for zoos about what to feed the mammals in their care / tour guide that will help people learn about what mammals they are likely to encounter when they go on holiday).

In order to find out information for the ('Mammal-Spotter's Guide' / feeding guide / tour guide), we're asking you to use a system called APECKS, which is supposed to be good at getting at people's knowledge. As you use the system, you'll build a record of what you know. Because the record is going to be used by other people, you should try to make things as clear as possible.

Knowledge engineers have developed ways of comparing what people know and of recording discussion, and during these sessions, you are going to be using one of these methods. This method involves having objects both summarising the differences between two peoples' knowledge and recording discussion about an area of knowledge. Have a look at the Vocabulary List and read the definitions for Comparison, Annotation and Criterion. You can refer to the vocabulary list throughout the sessions.

Everything that you do using this system will be logged for experimental use.

APECKS can be used through the World Wide Web, so you are going to be using Netscape. Please do not enter new URLs and please avoid using the 'Back' button where possible.

On the screen at the moment is a page which gives an overview of your Role.

Other people have also constructed roles, using the same individuals as you. Some of the roles have been made for different guides, so they have a different emphasis.

During this session, you will be looking at their roles, comparing them to yours, and making comments on them. You cannot change other people's roles, but you can make suggestions or comments on them.

Some of the roles might remind you of things or suggest to you information that you haven't put in your role. Also, other people can comment on what you've done, just as you can comment on theirs. You can reply to people's comments, or you can make changes to your role if you want.

You can look at an overview of the differences between your role and another person's using the 'Comparisons' button in the button bar of your role.

You can look at the comments made on any object using the 'Comments' button in the button bar of the object.

You can make comments on an object using the 'Annotate' button in the button bar of the object. Some prompts also encourage you to comment on your own and other peoples' roles.

If you think of a general principle or rule that summarises the decisions you made in detailing your role, you might want to make a criterion which summarises it. Then other

people can decide to use that criterion too. You can create criteria or add ones that other people have created from the Edit page of an object.

Please try to make as many useful comments as you can on your own and other people's roles, so that you can have a good discussion. Remember that people may answer your comments from one session to the next.

APPENDIX VIII MAMMALS

The mammals used in the study discussed in Chapter 7 were:

- Gorilla: *Gorilla gorilla*
- Eastern grey kangaroo: *Macropus giganteus*
- Polar bear: *Thalarctos maritimus*
- Grey squirrel: *Sciurus carolinensis*
- Indian elephant: *Elephas maximus*
- Sheep: *Ovis aries*
- Horse: *Equus caballus*
- Livingstone's flying fox: *Pteropus livingstonii*
- Sperm whale: *Physeter macrocephalus*
- Duck-billed platypus: *Ornithorhynchus anatinus*
- Western European hedgehog: *Erinaceus europaeus*
- Flying lemur: *Cynocephalus sp.*
- Three-toed sloth: *Bradypus tridactylus*
- Small-scaled tree pangolin: *Manis tricuspis*
- Rabbit: *Oryctolagus cuniculus*
- Brown rat: *Rattus norvegicus*
- Lion: *Panthera leo*
- Aardvark: *Orycteropus afer*
- Rock hyrax: *Procavia sp.*
- Caribbean manatee: *Trichechus manatus*
- Common tree shrew: *Tupaia glis*
- Sealion: *Zalophus californianus*
- Rufous elephant shrew: *Elephantulus rufescens*
- American black bear: *Ursus americanus*

APPENDIX IX LACEPEDE, 1799

The modern concept of the mammals - class Mammalia - as warm-blooded, hair-bearing animals which suckle their young with milk was established by 1758. However, in the 18th century biologists did not realise the importance of reproductive systems that are currently used to distinguish the monotremes (Prototheria), marsupials (Metatheria) and placental mammals (Eutheria). So early classifications, such as Lacepede (1799), are based on other characteristics - lumping all marine mammals together, distinguishing the winged bats as a separate group, and classifying all other mammals as quadrupeds, which are subdivided mainly on the basis of the structures of their hands and feet.

```

Mammals
  type*: True Quadrupeds
    type*: Digitigrades
      Grey squirrel
      Duck-billed platypus
      Three-toed sloth
      Rabbit
      Brown rat
    type*: Four-handed mammals
      Gorilla
    type*: One-hoofed mammals
      Horse
    type*: Pachyderms
      Indian elephant
    type*: Pedimanes
      Eastern grey kangaroo
    type*: Plantigrades
      Polar bear
      Western European hedgehog
      Lion
      American black bear
    type*: Two-hoofed mammals
      Sheep
  type*: Winged mammals
    Livingstone's flying fox
    Flying lemur
  type*: Marine mammals
    Sperm whale
    Caribbean manatee
    Sealion

```

APPENDIX X SIMPSON, 1945 (EXTANT GROUPS ONLY)

Simpson's (1945) classification of the mammals provides the yardstick for most recent classifications. It is based heavily on data discernible from the fossil record, and includes extinct taxonomic groups (not shown here).

Modern mammals are divided into three main categories on the basis of their reproductive biology: the monotremes (Prototheria), marsupials (Metatheria) and placental mammals (Eutheria). The monotremes are the only mammals that lay eggs; the marsupials give birth to very small offspring that further develop in a pouch; the placental mammals have offspring born at a more advanced state.

Simpson placed tree shrews amongst the Primates, but these are now placed in a distinct order (Scandentia). Simpson also placed elephant shrews amongst the Insectivora, but these are now placed in a distinct order (Macroscelidea).

```

Mammalia
  Subclass*: Prototheria
    Order*: Monotremata
      Duck-billed platypus
  Subclass*: Theria
    Infraclass*: Eutheria
      Order*: Artiodactyla
        Sheep
      Order*: Carnivora
        Polar bear
        Lion
        Sealion
        American black bear
      Order*: Cetacea
        Sperm whale
      Order*: Chiroptera
        Livingstone's flying fox
      Order*: Dermoptera
        Flying lemur
      Order*: Edentata
        Three-toed sloth
      Order*: Hyracoidea
        Rock hyrax
      Order*: Insectivora
        Western European hedgehog
        Rufous elephant shrew
      Order*: Lagomorpha
        Rabbit
      Order*: Perissodactyla
        Horse
      Order*: Pholidota
        Small-scaled tree pangolin
      Order*: Primates
        Gorilla
        Common tree shrew
      Order*: Proboscidea
        Indian elephant
      Order*: Rodentia
        Grey squirrel
        Brown rat
      Order*: Sirenia
        Caribbean manatee
      Order*: Tubilidentata
        Aardvark

```

Order*: Marsupialia
Eastern grey kangaroo

relationships between the various natural orders - it is a cladistic phylogeny. Although several recent attempts have been made to resolve the relationships within the placental mammals, there is still no single, unambiguous evolutionary tree.

APPENDIX XI BENTON, 1990

The classification scheme of Benton (1990) places more emphasis on the evolutionary relationships between the various mammal orders - it is a cladistic phylogeny. Although several recent attempts have been made to resolve the relationships within the placental mammals, there is still no single, unambiguous evolutionary tree.

```

Mammalia
  Infraclass: Theria
    Division*: Eutheria
      Cohort*: Edentata
        Order*: Xenarthra
          Three-toed sloth
        Order*: Pholidota
          Small-scaled tree pangolin
      Cohort*: Epitheria
        Superorder*: Anagalida
          Order*: Macroscelidea
            Rufous elephant shrew
          Order*: Rodentia
            Grey squirrel
            Brown rat
          Order*: Lagomorpha
            Rabbit
        Superorder*: Archonta
          Order*: Scandentia
            Common tree shrew
          Order*: Dermoptera
            Flying lemur
          Order*: Chiroptera
            Livingstone's flying fox
          Order*: Primates
            Gorilla
        Superorder*: Insectivora
          Western European hedgehog
        Superorder*: Ungulata
          Order*: Artiodactyla
            Sheep
          Order*: Cetacea
            Sperm whale
          Order*: Hyracoidea
            Rock hyrax
          Order*: Perissodactyla
            Horse
          Order*: Proboscidea
            Indian elephant
          Order*: Sirenia
            Caribbean manatee
        Superorder*: unnamed
          Order*: Carnivora
            Polar bear
            Lion
            Sealion
            American black bear
          Order*: Tubulidentata
            Aardvark
    Division*: Metatheria
      Order*: Marsupialia
        Eastern grey kangaroo
  Order: Monotremata
    Duck-billed platypus

```

APPENDIX XII CORBET & HILL, 1991

Corbet and Hill (1991) present a world list of currently living mammalian species. They are less interested in mammal classification above the taxon of genus. At the higher taxonomic level, their classification follows closely that of Simpson (1945), though they recognise the tree shrews (Scandentia) and the elephant shrews (Macroscelidea) as distinct orders. They also classify the Pinnipedia as a distinct order rather than as a suborder of the Carnivora.

Mammalia

- Order*: Monotremata
Duck-billed platypus
- Order*: Marsupialia
Eastern grey kangaroo
- Order*: Xenarthra
Three-toed sloth
- Order*: Insectivora
Western European hedgehog
- Order*: Scandentia
Common tree shrew
- Order*: Dermoptera
Flying lemur
- Order*: Chiroptera
Livingstone's flying fox
- Order*: Primates
Gorilla
- Order*: Carnivora
Polar bear
Lion
American black bear
- Order*: Pinnipedia
Sealion
- Order*: Cetacea
Sperm whale
- Order*: Sirenia
Caribbean manatee
- Order*: Proboscidea
Indian elephant
- Order*: Perissodactyla
Horse
- Order*: Hyracoidea
Rock hyrax
- Order*: Tubulidentata
Aardvark
- Order*: Artiodactyla
Sheep
- Order*: Pholidota
Small-scaled tree pangolin
- Order*: Rodentia
Grey squirrel
Brown rat
- Order*: Lagomorpha
Rabbit
- Order*: Macroscelidea
Rufous elephant shrew

APPENDIX XIII ZOOGEOGRAPHICAL REGIONS

The division of land masses into regions each with a characteristic fauna was first proposed in the mid 19th century, on the basis of the distribution of birds. These divisions, with some modifications, were found to apply also to reptiles and mammals. The zoogeographical regions owe the distinctness of their fauna to the barriers of dispersal to animals that have arisen during the Tertiary period (the last 66 million years).

Mammals

Nearctic

*Sealion**American black bear**Polar bear**Grey squirrel**Sheep*

Neotropical

*Three-toed sloth**Caribbean manatee*

Palearctic

*Polar bear**Grey squirrel**Sheep**Horse**Western European hedgehog**Rabbit*

Afrotropical

*Gorilla**Livingstone's flying fox**Rabbit**Lion**Aardvark**Rock hyrax**Rufous elephant shrew**Small-scaled tree pangolin*

Indomalayan

*Indian elephant**Flying lemur**Brown rat**Common tree shrew*

Australasian

*Eastern grey kangaroo**Duck-billed platypus*

Oceanic

*Sperm whale**Sealion*